Resum

L'objectiu d'aquest projecte és realitzar el control d'un dron quadricòpter (el model AR.Drone de l'empresa Parrot) fent servir el programa SimuLink integrat en la *suite* matemàtica MatLab, a fi que se li puguin programar trajectòries i consignes de velocitat des de un ordinador.

Per a tal efecte, aquest treball s'estructura en dues fases. En la fase inicial, es treballa amb un simulador que fa servir un model matemàtic del dron, el "ARDroneKit", que és un paquet de SimuLink ja preparat amb tot el necessari per a treballar. Es modelitzen dos tipus de controladors diferents: un per realimentació d'estat (integrant també un observador d'estat) i un controlador PID. Després de descriure tot el procés de càlcul dels diferents elements de cada tipus de controlador, aquests s'apliquen al model emprant unes consignes de prova preparades prèviament, i es comprova la seva efectivitat en el control del model.

Essent satisfactori el resultat de les simulacions, aleshores, aquests mateixos controladors passen a aplicar-se a un dron real, fent servir el mateix paquet de SimuLink, que permet connectar-se als sensors del dron i enviar-li les accions de control que es calculin des del controlador.

En primer lloc, s'aplica el controlador a cadascun dels subsistemes d'interès per al projecte (velocitats frontal i lateral, orientació i altura), en els quals es valida per separat, perquè el sistema global no es descontroli quan s'intentin manipular diverses variables alhora. Quan aquests es determinen com a correctes, es procedeix a realitzar el control d'un vol en cercles amb l'aparell, controlant tots aquests aspectes simultàniament, per comprovar el seu funcionament.

Finalment, el controlador PID (que és l'únic que es va poder provar en experimentació real) rep uns últims ajustos i es dóna com a vàlid per a realitzar el control, tenint en compte, però, algunes consideracions, entre les que destaca la imprecisió d'alguns sensors integrats en el dron i la millora que suposaria aplicar-hi sensorització externa.

Sumari

1 PREFACI

En aquests últims anys hi ha hagut un creixement molt pronunciat en el mercat dels drons quadricòpters. Aquests són uns aparells voladors que s'eleven mitjançant quatre hèlices i permeten controlar-los com si es tractés d'un cotxe de radio-control. No obstant, l'autèntic *boom* es va produir quan l'empresa Parrot va presentar el seu AR.Drone el 2010, que va acostar el públic general a aquest món, amb un funcionament senzill i uns preus assequibles.

Aquest aparell disposa d'una sèrie de sensors que permeten el seu control [1]. Entre ells, destaquen:

- Dues càmeres, una frontal i una vertical, que permeten tenir el punt de vista de l'aparell en dues direccions. La frontal permet controlar el dron durant el vol, mentre que la vertical serveix per als aterratges.
- Una IMU (*Inertial Measurement Unit*) que permet mesurar la velocitat de l'aparell i, juntament amb un magnetòmetre, determinar la seva orientació en el pla.
- Un altímetre de tipus sònar.

Fent servir tota aquesta informació, és necessari fer que l'aparell realitzi el vol seguint una trajectòria determinada, a fi que no vagi a llocs no desitjats o impacti amb obstacles que el puguin fer malbé. Cal, aleshores, disposar d'un mecanisme de control sobre ell.

Aquest control es pot fer de dues maneres. Per una banda, l'app oficial per a iOS i Android connecta amb l'aparell, i permet fer-ne un pilotatge en temps real, mentre es veu la imatge de les càmeres per pantalla.

En l'altra opció, un PC pot connectar-se al dron, llegir els seus sensors i, havent programat una trajectòria prèviament, calcular quines correccions cal fer en el vol per enviar-les de tornada a l'aparell.

1.2 Antecedents

Dins de l'àmbit de la UPC, ja s'han presentat amb anterioritat projectes relacionats amb els drons, principalment entre els estudiants d'Enginyeria Aeronàutica. Han estat útils en l'elaboració d'aquest projecte per prendre com a referència d'alguns aspectes.

En concret, s'ha tingut accés als projectes del Marcel Soler Ferret (*Estudi del control de trajectòria d'un UAV multirotor amb Robotic Operating System*), del Sergi Costal Acosta (*Estudi d'algoritmes de prognosi aplicats a UAVs multirotors*) i al del Carlos Alcaraz Lara (*Estudi d'identificació d'un model matemàtic per a la realització del control de trajectòria d'un quadricòpter*).

2 INTRODUCCIÓ

2.1 Motivacions

Com a estudiant d'enginyeria, des de petit he tingut inquietud per saber com funcionen les coses. En concret, la robòtica és un tema que sempre m'ha cridat l'atenció, i el *boom* d'aquests últims anys en el que respecta a drons quadricòpters (en concret del Parrot AR.Drone) em va deixar amb ganes de pilotar-ne un. Per això, quan va arribar l'hora de decidir el tema del Projecte de Final de Carrera, vaig pensar que seria interessant treballar amb un d'aquests aparells, conèixer-lo per dins i poder controlar-lo jo mateix.

2.2 Objectius del projecte

L'objectiu principal d'aquest projecte és que en acabar es disposi d'un controlador funcional implementat en MatLab que sigui capaç de controlar els moviments d'un d'aquests drons, amb la finalitat de poder programar-hi diferents trajectòries i comanes.

Aquest objectiu el componen uns altres dos, que són ben diferenciats, i cadascun correspon a una de les dues fases del projecte. En la primera no es farà servir un dron real, s'emprarà un simulador, a fi que les primeres proves (les quals acostumen a presentar errors) no puguin fer malbé l'aparell. En aquestes, l'objectiu serà establir alguns mecanismes de control previs per al dron, fent servir diferents sistemes d'observació i control. Un cop el simulador doni uns resultats satisfactoris, es pot procedir a treballar amb el dron real.

En la segona fase, es fan servir els controladors obtinguts a través de la simulació en el dron, i l'objectiu és el d'acabar d'ajustar els controladors obtinguts prèviament. En primer lloc, aquests es validaran per a cada tipus de moviment per separat (frontal, lateral, gir i altura) per, finalment, manipular tots aquests aspectes alhora aconseguint que l'aparell es comporti de la forma desitjada.

2.3 Abast

El projecte consisteix en elaborar un mecanisme de control per al dron amb un PC. Per tant, el dron no és modificat en absolut, tant a nivell hardware com software. Tenint els controladors preparats, el projecte es pot replicar amb un aparell acabat de treure de la caixa.

Per a fer la simulació s'emprarà el programa SimuLink, integrat dins la suite matemàtica MatLab, en la seva versió R2013b. Es farà servir un paquet preparat d'aquest, l'anomenat "ARDroneKit", el qual ja integra un model matemàtic del dron, així com la configuració prèvia perquè funcioni correctament i un mecanisme de control llest per funcionar des de l'inici. Durant el transcurs del projecte, aquest s'elimina i es modifica per el dissenyat, de forma que finalment només es manté de l'original el nucli principal del model, havent canviat tota la resta.

3 MODELITZACIÓ I SIMULACIÓ

En primer lloc, és necessari veure com s'estructura el model de SimuLink que es fa servir, a fi d'entendre el seu funcionament i saber com manipular-lo.

Aquest es basa en un conjunt de variables molt utilitzat en aviació, que determinen l'orientació de l'aparell en els tres eixos de l'espai. La variable *Roll*(φ) mesura la inclinació lateral del dron, positiva quan s'inclina cap a la dreta. De manera similar, el *Pitch*(θ) mesura la inclinació frontal de l'aparell, sent positiva quan aquest apunta cap amunt. El *Yaw*(ψ), en canvi, mesura l'orientació del dron en el pla horitzontal, determinant com a angle positiu un gir cap a l'esquerra (sentit antihorari).

Paral·lelament, es fan servir les velocitats de l'aparell, tant en la seva direcció frontal (U) com lateral (V). Aquestes es mesuren en el sistema de referència del dron i no en el fix al terra, ja que això requereix sensorització externa. Per últim, també es pren com a variable l'altura fins al terra (z).

Aquestes sis són les variables de sortida de cadascun dels subsistemes en els que el dron es divideix. No obstant, aquests requereixen d'unes variables d'entrada a les quals respondre. Aquest vector de variables està constituït per quatre, i són el *Roll* de referència (φ_{ref}), el *Pitch* de referència (θ_{ref}), la derivada del *Yaw* (velocitat de gir, $\dot{\psi}_{ref}$) i la derivada de l'altura (velocitat d'elevació $\dot{z}_{ref}$).

Cadascun dels subsistemes que conforma el model de treball té una finalitat, i entrega una variable de sortida diferent, prenent alguna de les d'entrada. En la taula següent es detallen els noms de cadascun, així com les seves variables corresponents:

	Var. d'entrada	Var. de sortida
$tfRollAng3$	φ_{ref}	φ
$tfRoll2V$	φ_{ref}	V
$tfPithAng3$	θ_{ref}	θ
$tfPitch2U$	θ_{ref}	U
$tfYaw$	$\dot{\psi}_{ref}$	ψ
tfH	$\dot{z}_{ref}$	z

El funcionament de cadascun d'aquests subsistemes està governat per una funció de transferència, la qual determina el seu comportament i característiques enfront de les diferents variacions a l'entrada. Aquestes són:

$$tfRollAng3 = \frac{1{,}483s + 3{,}524}{s^2 + 4{,}268s + 12{,}69}, \qquad tfRoll2V = \frac{4{,}774}{s + 0{,}4596}$$

$$tfPitchAng3 = \frac{2{,}514s + 4{,}866}{s^2 + 3{,}978s + 11{,}92}, \qquad tfPitch2U = \frac{-6{,}154}{s + 0{,}665}$$

$$tfYaw = \frac{1{,}265}{s + 0{,}005879}, \qquad tfH = \frac{0{,}1526s + 5{,}153}{s^2 + 5{,}82s + 1{,}375 \cdot 10^{-11}}$$

Com es pot apreciar, aquests subsistemes del model són lineals. Això significa que davant una variació k de l'entrada, es comporten de manera proporcional, i que davant una suma de dos valors a l'entrada, la sortida és també la suma de les respostes a cadascuna.

Aquest model, però, també es pot treballar en forma d'espai d'estats. D'aquesta manera, s'amplia el control no només a les variables de sortida, sinó a les seves variables internes. Per a fer servir un controlador per realimentació d'estat i el seu observador corresponent el canvi és necessari, ja que requereixen de les matrius que constitueixen el model en aquesta forma.

Fent la conversió de cadascun amb la comana `tf2ss` de MatLab, el model queda de la següent manera:

- $tfRollAng3$ passa a ser $ssRoll$:

$$A = \begin{pmatrix} -4{,}268 & -3{,}172 \\ 4 & 0 \end{pmatrix};\ B = \begin{pmatrix} 2 \\ 0 \end{pmatrix};\ C = (0{,}7417 \quad 0{,}4405);\ D = (0)$$

- $tfRoll2V$ passa a ser $ssRoll2V$:

$$A = (-0{,}4596);\ B = (2);\ C = (2{,}387);\ D = (0)$$

- $tfPithAng3$ passa a ser $ssPitch$:

$$A = \begin{pmatrix} -3{,}978 & -2{,}98 \\ 4 & 0 \end{pmatrix};\ B = \begin{pmatrix} 2 \\ 0 \end{pmatrix};\ C = (1{,}257 \quad 0{,}6083);\ D = (0)$$

- $tfPitch2U$ passa a ser $ssPitch2U$:

$$A = (-0{,}665);\ B = (2);\ C = (-3{,}077);\ D = (0)$$

- $tfYaw$ passa a ser $ssYaw$:

$$A = (-0{,}005879); B = (1); C = (1{,}265); D = (0)$$

- tfH passa a ser ssH:

$$A = \begin{pmatrix} -5{,}82 & -3{,}605 \cdot 10^{-6} \\ 3{,}815 \cdot 10^{-6} & 0 \end{pmatrix}; B = \begin{pmatrix} 1024 \\ 0 \end{pmatrix}; C = (0{,}0001491 \quad 1319); D = (0)$$

No obstant, tot i que originàriament el model del paquet "ARDroneKit" és lineal, doncs fa servir aquestes mateixes funcions de transferència, el model de treball ha estat modificat prèviament per a incloure unes no-linealitats afegides, de forma que es representi més adequadament una situació real, així com els sorolls introduïts pels sensors en les mesures, la qual cosa genera imprecisions i errors de lectura. En l'apartat 3.2 s'explica el procediment de treball per tenir en compte aquests efectes no-lineals i incerteses.

3.1 Control en llaç obert

L'objectiu del projecte és realitzar un control en llaç tancat implementat en un simulador, per després realitzar el control del dron real. Tot i això, abans d'arribar-hi, és interessant veure el seu comportament en llaç obert, és a dir, calculant les entrades exactes perquè aquest es comporti de la manera desitjada, sense llegir el seu estat per a res. Aquest tipus de control és molt senzill d'implementar, però té el clar inconvenient de no corregir cap tipus d'error que es presenti durant l'execució.

Per a realitzar aquesta part prèvia del projecte, es prepara una trajectòria a seguir, a la qual es va ajustant el comportament del dron mitjançant un procediment d'assaig-error. Es proven primer unes senyals inicials en les entrades dels subsistemes, i es van modificant fins que el resultat és el desitjat. La trajectòria que es vol realitzar (amb una espera de 2 segons entre punts) és la següent, partint del repòs $(0,0,0)$:

- Enlairar-se (posició $(0,0,1)$)
- Moure's a $(5,0,1)$
- Moure's a $(5,5,1)$
- Girar 135º per encarar el punt $(0,0,1)$
- Moure's a $(0,0,1)$ (en diagonal, moviment de longitud $\sqrt{50} = 7{,}07$)
- Aterrar $(0,0,0)$

Per a això, s'agafa el model del dron i se li treu tots els mecanismes de control, deixant lliures les entrades al sistema per a introduir-hi posteriorment les consignes necessàries. El seu aspecte és aquest:

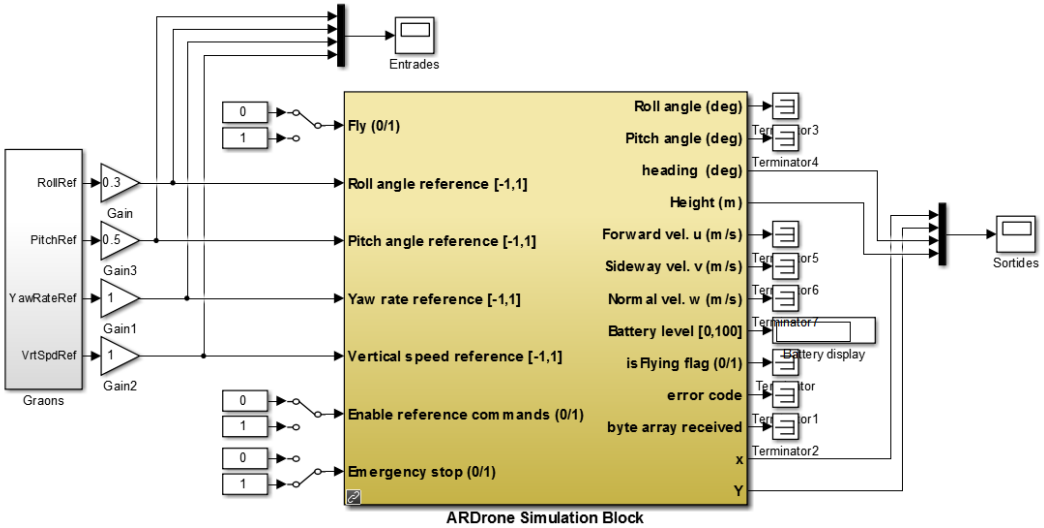


Fig. 3.1: Model del sistema en llaç obert.

Les consignes que es fan servir tenen forma de graons unitaris, els quals passen per uns blocs de guany ja presents en el model original per regular la seva amplitud, a fi que no saturin les entrades al sistema. El procediment consisteix en aplicar un graó unitari que després és anul·lat al cap d'un cert temps per un graó oposat, de forma que el control es realitza ajustant aquest temps entre l'un i l'altre.

Després d'un procés d'assaig-error mesurant la posició del dron, els graons que resulten en la trajectòria desitjada són els següents:

- Entrada "*Roll angle reference*":

$$u_1(t) = U(t-14) - U(t-(14+3))$$

- Entrada "*Pitch angle reference*":

$$u_2(t) = U(t-6) - U\big(t-(6+1{,}8)\big) + U(t-26) - U\big(t-(26+2{,}5)\big)$$

- Entrada "*Yaw rate reference*":

$$u_1(t) = U(t-22) - U(t-(22+1{,}89))$$

- Entrada "*Vertical speed reference*":

$$u_2(t) = U(t-2) - U\big(t-(2+1{,}15)\big) + U(t-36) - U\big(t-(36+1{,}15)\big)$$

L'aspecte visual de les diferents consignes, un cop adequats els seus valors amb els guanys anteriorment citats, és aquest:

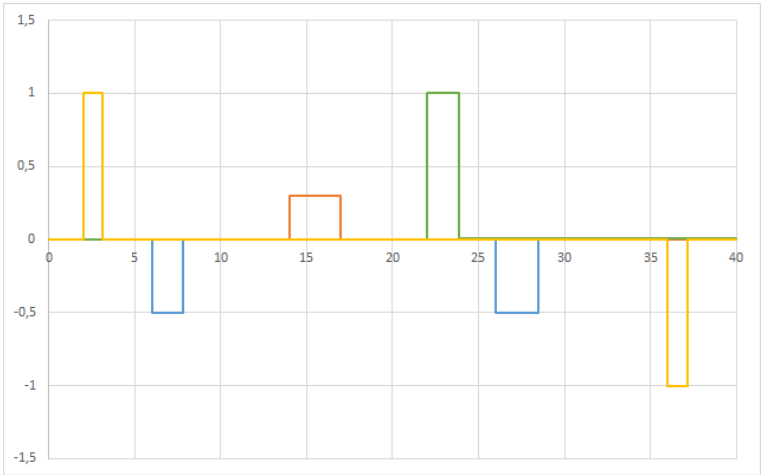
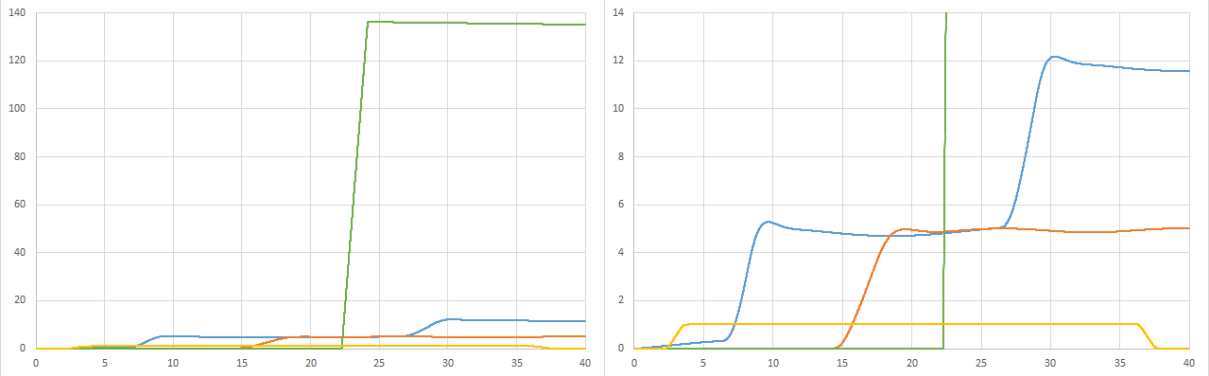


Fig. 3.2: Consignes aplicades a l'entrada del sistema: Pitch ref. *(blau)*, Roll ref. *(taronja)*, Yaw rate ref. *(verd) i* Vertical speed ref. *(groc). Eix horitzontal en segons.*

I la resposta que aquestes generen sobre la posició (x, y, z) del dron es presenta en la gràfica següent. No obstant, com que la posició es calcula mitjançant l'integral de les velocitats frontal i lateral, aquesta està donada en el sistema de referència del dron, no en la referència del terra:



Figs. 3.3 i 3.4: Evolució del desplaçament frontal (blau), lateral (taronja), del Yaw *(verd) i de l'altura (groc), amb una segona gràfica ampliada degut a les diferències d'escala entre el* Yaw *i la resta de variables. Mesures en metres per a les posicions i l'altura, en graus per al* Yaw*, eix horitzontal en segons.*

En primer lloc, es pot veure com l'altura augmenta fins a un valor de 1 m. Tot seguit, la posició frontal creix fins a un valor de 5 m, seguida de la lateral, que també puja fins a aquest valor. Aleshores, el *Yaw* creix fins al valor de 135º i el dron queda encarat amb l'origen, cap al qual avança fent un desplaçament frontal de $\sqrt{50}$ m (aproximadament 7,07 m), que és la longitud de la hipotenusa del triangle de catets 5 i 5, fins a quedar en un valor aproximat de 12 m. Finalment, l'altura torna al seu valor inicial de 0 m.

Tot i que aproximadament el sistema es situa als valors desitjats, s'aprecia que aquest no està completament quiet, oscil·la al voltant dels punts en què s'ha detingut, degut a diverses no-linealitats presents en el model, el soroll generat pels sensors del dron real, així com diferents tipus de pertorbacions i retards. En l'apartat 3.2 es procedeix a convertir-lo en un sistema lineal, a fi de fer les primeres proves amb menys dificultat, per després tornar a treballar amb el model complet.

3.2 Simplificació del sistema per al seu tractament

Els sistemes reals poden presentar de vegades valors bastant singulars. En alguns casos, un valor molt proper a zero en algun dels pols d'un sistema pot provocar errors no desitjats, així com fer que el model es descontroli. En canvi, si aquest valor s'ajusta manualment (per exemple, es posa a zero), el simulador hi treballa molt més fàcilment.

El model de treball d'aquest projecte és un d'aquests casos. El major problema ve donat per un dels pols del sistema tfH/ssH, ja que en té un a -5.82, mentre que l'altre està situat a $-2.36 \cdot 10^{-12}$. Aquest pol tan petit fa que el sistema es comporti pràcticament com si tingués un integrador pur (pol a zero), però a l'hora de fer els càlculs apareixen matrius amb valors molt exagerats que provoquen errors i inestabilitats en el sistema.

Per solucionar-ho, es crea un sistema modificat $tfHM$, que sí tindrà aquest integrador pur i es farà servir per realitzar els càlculs. Posteriorment, els controladors resultants s'aplicaran al model original, per realitzar-ne el control. Es crea un petit *script* de MatLab per a fer aquest canvi:

```
tfHM=tfH;
tfHM.den=[1 5.82 0];
ssHM=ss(tfHM)
```

El model resultant és un amb números més senzills, bastant més que els de tfH:

$$tfHM = \frac{0{,}1526s + 5{,}153}{s^2 + 5{,}82s}$$

$$A = \begin{pmatrix} -5{,}82 & 0 \\ 1 & 0 \end{pmatrix}; B = \begin{pmatrix} 2 \\ 0 \end{pmatrix}; C = \begin{pmatrix} 0{,}07632 & 0{,}576 \end{pmatrix}; D = (0)$$

Una altra simplificació que cal fer també consisteix a fer una aproximació prèvia dels controladors fent servir un model completament lineal (sense soroll ni retards) a fi de simplificar els càlculs. Per aconseguir-ho, aquest nou model es basa únicament en les equacions del sistema donades a l'inici, convertides a un bloc de Simulink amb una entrada i una sortida cadascuna.

Posteriorment, els resultats obtinguts s'introdueixen al model no-lineal complet i se'ls fan els ajustos necessaris perquè funcionin correctament amb aquest.

3.3 Creació d'una consigna adequada

Com a últim pas abans de començar a desenvolupar els diferents controladors, per avaluar el correcte funcionament del sistema realimentat és convenient crear una consigna que sigui fàcilment comparable amb les sortides del sistema, de forma que la coincidència entre elles indica un bon disseny d'aquests mecanismes de control.

Una primera opció és fer que les consignes siguin totes zero, mentre que l'estat inicial del sistema es programa a uns valors diferents. Si controlador és correcte, quan la simulació s'iniciï els valors de cada variable de sortida han d'anar-se acostant progressivament a zero.

Una altra és dissenyar consignes que puguin ser identificades fàcilment amb la forma de la sortida del sistema. Això vol dir que s'han de poder veure els errors que apareguin a la sortida a simple vista. Un exemple són les funcions sinusoïdals. Aplicant-les aquestes tant a la velocitat frontal com a la lateral s'aconsegueix un moviment circular. La seva forma és:

$$U(t) = A \cdot \sin(\omega t)$$

$$V(t) = A \cdot \cos(\omega t) = A \cdot \sin(\omega t + \frac{\pi}{2})$$

on l'amplitud A indica el valor màxim de cadascuna de les velocitats de moviment, mentre que la velocitat angular ω indicaria la seva freqüència d'oscil·lació.

Per als sistemes $ssYaw$ i ssH, és més habitual que funcionin a base de graons, la consigna per a aquests es programa com una sèrie de polsos.

En conjunt, l'aspecte de totes aquestes diferents consignes és el següent:

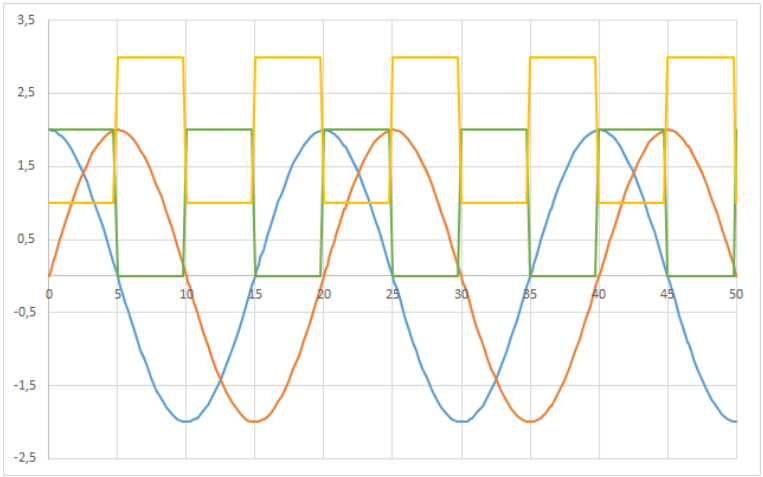


Fig. 3.5: Consigna de prova del model, amb les velocitats lateral i frontal representades en blau i taronja respectivament, el Yaw *en verd i l'altura en groc. Les mesures estan en metres per segon, radians i metres, respectivament, eix horitzontal en segons.*

3.4 Control en llaç tancat per realimentació d'estat

Un cop que la consigna de prova està determinada, es procedeix a aplicar al model el mecanisme de control en llaç tancat. Se'n realitzen dos diferents, un fent servir un realimentador d'estat, el qual s'estudia i dissenya en aquest apartat, i un controlador PID, que es presenta a l'apartat 3.5.

Aquest és el diagrama general d'aquest primer mecanisme de control, del qual tot seguit s'explica el seu funcionament:

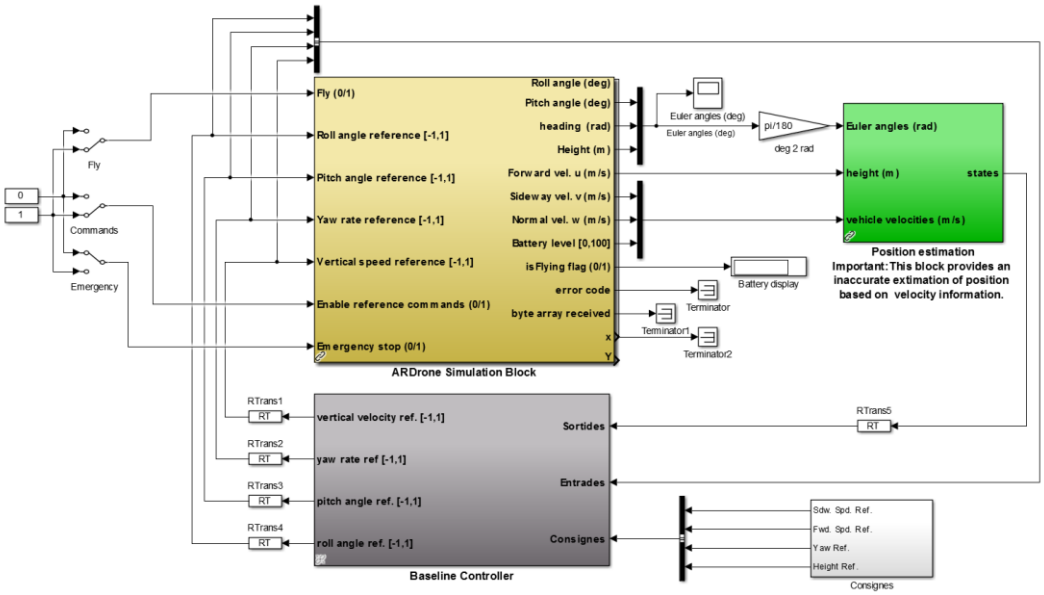


Fig. 3.6: Model del sistema en llaç tancat, amb un realimentador d'estat (bloc gris).

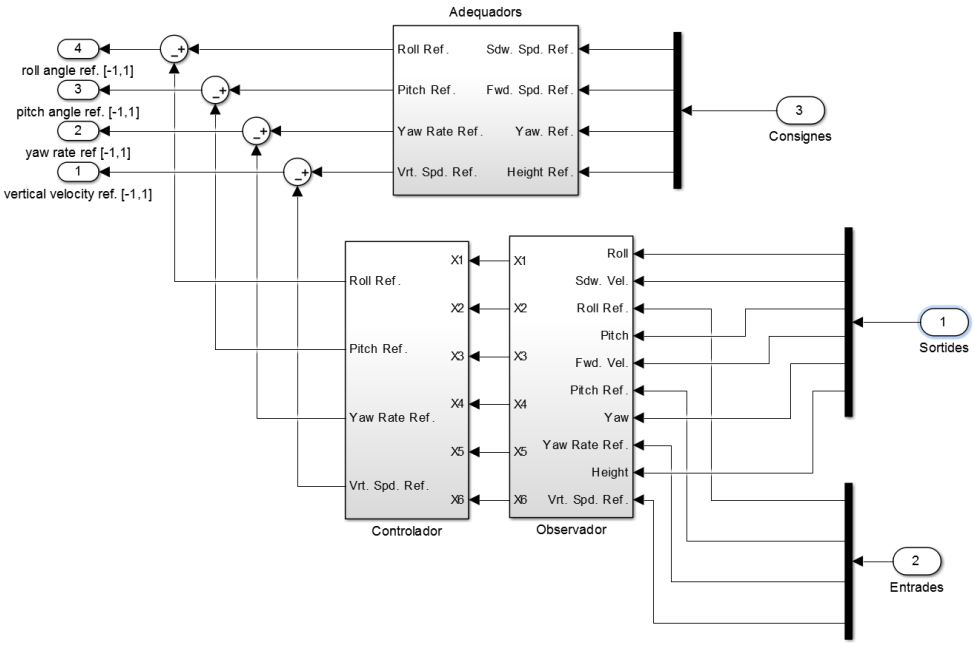


Fig. 3.7: Estructura interna del controlador per realimentació d'estat (bloc gris de la Fig. 3.6). S'hi poden veure les tres parts principals que el composen: observador, controlador i adequador de consigna.

Per a implementar aquest sistema de control cal treballar amb la representació en espai d'estats del model en comptes d'amb les funcions de transferència, ja que el que es busca estabilitzar no són les sortides, sinó les variables internes que les generen. No obstant, aquesta informació no és disponible directament des dels sensors.

Primer de tot, el bloc de color verd de la Fig. 3.6 agafa les diferents variables d'estat del sistema i les agrupa en un únic vector. Tot seguit, aquest vector entra al bloc de control (el de color gris). Allà, L'observador d'estat l'agafa, juntament amb el vector d'entrades al sistema, per fer una estimació de les variables d'estat internes (les variables X), les quals s'introdueixen al controlador.

Aleshores, el bloc controlador les multiplica per una matriu calculada a tal efecte, i resta el resultat del vector de consignes per a produir les accions de control necessàries per a que el sistema respongui correctament. Aquest vector de consignes, però, passa prèviament per un bloc adequador que n'ajusta el seu valor, fent-les comparables a les senyals presents a la sortida del controlador.

Abans de fer cap càlcul, però, cal saber si el control mitjançant aquest mecanisme és viable. El procés consisteix en analitzar la controlabilitat i observabilitat, dues propietats que el sistema ha de complir, i si el resultat és favorable aleshores prosseguir. Aquesta anàlisi es realitza en el subapartat 3.4.1, per posteriorment passar a la creació i càlcul d'aquests tres blocs del mecanisme de control (observador, controlador i adequador) els quals es detallen en els subapartats 3.4.2, 3.4.3 i 3.4.4 d'aquest mateix capítol, respectivament.

3.4.1 Anàlisi de controlabilitat i observabilitat

Abans de començar el càlcul del controlador per realimentació d'estat, tal com s'exposa al final de l'apartat 3.4, cal determinar si aquest mecanisme és viable. Per tant es passa a comprovar la controlabilitat i observabilitat dels subsistemes que conformen el model.

L'explicació general d'aquestes dues propietats és senzilla [3]. En concret, que un sistema sigui controlable implica que des de qualsevol estat inicial, aquest pot ser portat a un valor arbitrari mitjançant unes determinades accions de control. D'altra banda, que un sistema sigui observable vol dir que es pot conèixer el seu estat a partir les dades de les sortides de què es disposa, així com de les entrades aplicades a aquest.

Tot seguit es descriu el procediment d'avaluació d'aquestes dues propietats. Partint d'un sistema genèric, amb n variables d'estat, r entrades i m sortides:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix};\ B = \begin{pmatrix} b_{11} & \cdots & b_{1r} \\ \vdots & \ddots & \vdots \\ b_{n1} & \cdots & b_{nr} \end{pmatrix};\ C = \begin{pmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{r1} & \cdots & c_{rn} \end{pmatrix};\ D = \begin{pmatrix} c_{11} & \cdots & c_{1r} \\ \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{mr} \end{pmatrix}$$

L'avaluació de la controlabilitat es realitza en dos passos [3]. En primer lloc, es construeix la matriu de controlabilitat wc, per a la qual es pren inicialment la matriu B, se li adjunta per la dreta el producte d'AB, després A^2B, ..., fins $A^{n-1}B$. Aleshores, es van prenent diferents columnes d'aquesta que siguin linealment independents entre sí, tractant d'arribar a una quantitat n de columnes, la qual cosa implica que aquesta matriu wc té rang n. Si això es compleix, aleshores el sistema és controlable.

El procés per avaluar l'observabilitat és molt similar al de la controlabilitat [3]. En aquest cas, es construeix la matriu d'observabilitat wo prenent inicialment la matriu C, i se li adjunten per sota els productes CA, CA^2, ..., fins CA^{n-1}. De forma semblant, es prenen fileres linealment independents fins a arribar a una matriu de dimensió n. Si és possible fer-ho, el sistema és, llavors, observable.

No obstant, el programa MatLab té dues comanes que generen automàticament les matrius wc i wo, restant només fer la comprovació del rang d'aquestes, que ha de ser igual al nombre de variables d'estat que tingui cada subsistema del model. Aquestes comanes són `ctrb` i `obsv` respectivament, i a l'annex A.1 es detalla el *script* escrit per construir les matrius i fer les comprovacions de rang per a cadascuna, i que mostra finalment per pantalla si cadascun d'ells és controlable i/o observable.

Tot seguit es mostra el resultat d'aplicar aquest procediment, descrivint les diferents matrius wc i wo que, donat que els subsistemes posseeixen 1 ó 2 variables d'estat cadascun, una entrada i una sortida, tenen dimensió $1x1$ ó $2x2$:

$$wcRoll = \begin{pmatrix} 2 & -8{,}5366 \\ 0 & 8 \end{pmatrix}, \qquad woRoll = \begin{pmatrix} 0{,}7417 & 0{,}4405 \\ -1{,}4039 & -2{,}3524 \end{pmatrix}$$

$$wcRoll2V = (2), \qquad woRoll2V = (2{,}3868)$$

$$wcPitch = \begin{pmatrix} 2 & -7{,}9568 \\ 0 & 8 \end{pmatrix}, \qquad woPitch = \begin{pmatrix} 1{,}2569 & 0{,}6083 \\ -2{,}5675 & -3{,}7451 \end{pmatrix}$$

$$wcPitch2U = (2), \qquad woPitch2U = (-3{,}0772)$$

$$wcYaw = (1), \qquad woYaw = (1{,}2653)$$

$$wcH = 1000 \cdot \begin{pmatrix} 1{,}024 & -5{,}9597 \\ 0 & 3{,}9063 \cdot 10^{-6} \end{pmatrix}, \qquad woH = 1000 \cdot \begin{pmatrix} 1{,}4907 \cdot 10^{-7} & 1{,}3191 \\ 4{,}1645 \cdot 10^{-6} & -5{,}37 \cdot 10^{-13} \end{pmatrix}$$

No obstant, tal com es discuteix en l'apartat 3.2, les matrius wcH i woH presenten uns valors molt dispars, amb molts ordres de magnitud entre ells, la qual cosa ve generada per el pol del sistema ssH en $-2.36 \cdot 10^{-12}$. Aleshores, també s'aplica el mateix *script* al sistema modificat $ssHM$, per veure si el canvi del pol a zero simplifica les matrius corresponents:

$$wcHM = \begin{pmatrix} 2 & -11{,}64 \\ 0 & 2 \end{pmatrix}, \qquad woHM = \begin{pmatrix} 0{,}0763 & 2{,}5765 \\ 2{,}1322 & 0 \end{pmatrix}$$

Efectivament, els valors presents en el sistema $ssHM$ no tenen tants ordres de magnitud entre ells.

Com a conclusió d'aquest procediment, es determina que tots els subsistemes són tant controlables com observables i que, per tant, es pot procedir a realitzar els càlculs pertinents per a calcular el sistema de realimentació d'estat.

3.4.2 Disseny del controlador per realimentació d'estat

Tenint la seguretat que el sistema compleix les dues condicions de l'apartat 3.4.1, és necessari dissenyar un controlador per a aconseguir un comportament adequat.

Per a fer-ho, el controlador agafa les variables internes d'estat i calcula l'acció de control necessària, retornant-la de nou al sistema per les seves entrades. No obstant, no es disposa de les variables internes d'estat d'una forma directa. La solució passa per dissenyar un observador d'estat (calculat en l'apartat 3.4.3), el qual entrega una estimació d'aquestes, que a efectes del controlador és com si es tractés de les variables reals.

El càlcul d'aquest tipus de controladors es fa per assignació de pols [3]. Partint d'un sistema genèric (matrius A, B, C, D) els valors propis de la matriu A (que al mateix temps són el valor dels pols del sistema) són els que determinen el comportament global d'aquest: els pols són estables si tenen part real negativa i el sistema ho és si tots els pols ho són. Si algun d'ells no fos estable, aleshores és necessari realimentar l'estat de forma que, passant per una matriu de control K, s'apliqui la correcció adequada al sistema. Aquest és el diagrama d'un sistema d'aquesta mena:

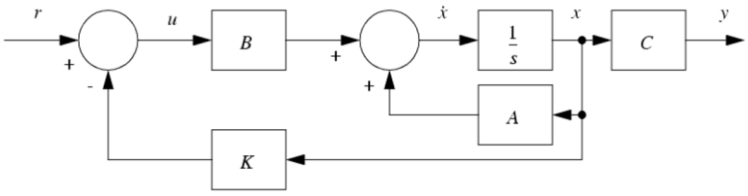


Fig. 3.8: Diagrama de blocs d'un sistema genèric amb realimentació d'estat.

En recalcular les equacions d'estat del sistema després d'afegir la matriu K s'arriba al següent:

$$\dot{x}(t) = (A - BK)x(t) + Br(t)$$

Ara, els valors propis de la dinàmica d'aquest sistema (que en aquest moment depenen dels valors encara no determinats de K) s'han d'igualar amb el conjunt de valors propis desitjats per al sistema realimentat (estables o, si ja ho eren, més ràpids que abans), i a partir d'aquestes igualtats poder determinar els valors de K.

Per a fer aquest càlcul ràpidament, MatLab disposa d'una comana anomenada `place`. Aquesta pren com a paràmetres les matrius A i B, així com el conjunt de valors propis desitjats, i retorna la matriu K corresponent al sistema realimentat. El *script* que realitza el càlcul del controlador per a cada subsistema es troba en l'annex A.2.

És necessari, llavors, determinar quins pols es vol que tingui el sistema realimentat. Primer de tot, cal saber quins són els pols actuals de cadascun dels subsistemes del model. Per a això, es fa servir la comana `eig` de MatLab, la qual retorna els valors propis d'una matriu (que en el cas de la matriu A, són els pols del sistema mateixos). El resultat d'aplicar-la a cadascun és el següent:

$$eig(ssRoll.A) = \begin{pmatrix} -2.1342 + 2.8517i \\ -2.1342 - 2.8517i \end{pmatrix}, \qquad eig(ssRoll2V.a) = -0.4596$$

$$eig(ssPitch.a) = \begin{pmatrix} -1.9892 + 2.8216i \\ -1.9892 - 2.8216i \end{pmatrix}, \qquad eig(ssPitch2U.a) = -0.6650$$

$$eig(ssYaw.a) = -0.0059, \qquad eig(ssHM.a) = \begin{pmatrix} 0 \\ -5.82 \end{pmatrix}$$

L'objectiu és que els sistemes resultants siguin estables, osigui que és necessari que els pols de cadascun tinguin tots part real negativa (i, a més, no massa propera a zero). Tenint això en compte, per als sistemes $ssRoll2V$, $ssPitch2U$ i $ssYaw$ s'han imposat pols de valor -2. Per al sistema $ssHM$, com que és d'ordre 2 i dos pols exactament iguals poden donar problemes amb el procés de càlcul de MatLab, s'han imposat els pols a -2 i -2.01. Finalment, com els pols dels sistemes $ssRoll$ i $ssPitch$ ja tenen part real propera a -2, s'han imposat a un valor igual al doble del que tenen ara, a fi de fer-los més ràpids. Les matrius de control resultants per a cada subsistema són:

$$K_{Roll} = \begin{pmatrix} 19{,}2074 \\ 156{,}9956 \end{pmatrix}, \qquad K_{Roll2V} = 0{,}7702$$

$$K_{Pitch} = \begin{pmatrix} 17{,}9027 \\ 147{,}4902 \end{pmatrix}, \qquad K_{Pitch2U} = 0{,}6675$$

$$K_{Yaw} = 1{,}9941, \qquad K_{HM} = \begin{pmatrix} -0{,}9050 \\ 2{,}01 \end{pmatrix}$$

3.4.3 Disseny de l'observador d'estat

Tot seguit, després del càlcul del controlador, cal introduir-li les variables d'estat del sistema, a fi que pugui realitzar la seva feina. No obstant, aquestes no acostumen a ser llegibles directament des de les sortides, de forma que es requereix un intermediari que, prenent els valors de les sortides i de les entrades del sistema, sigui capaç de fer una estimació d'aquestes variables inaccessibles, i que el controlador les pugui fer servir com si fossin les originals. Aquesta és la tasca de l'observador d'estat, també anomenat estimador.

Cal fer aleshores l'elecció d'un tipus d'observador. En aquest cas, s'escull un d'ordre complet per al projecte. La raó de triar aquest enfront d'un d'ordre reduït és que al tenir poques variables d'estat (1 ó 2, depenent del subsistema) un observador reduït dificultaria el càlcul, ja que són més adients per a nombres de variables més grans. Presenta aquest diagrama de blocs:

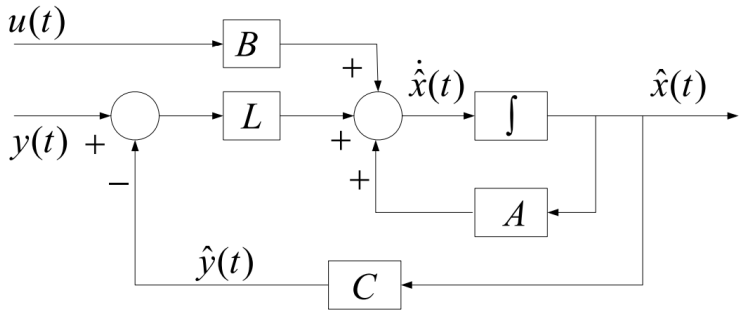


Fig. 3.9: Diagrama de blocs d'un observador d'estat d'ordre complet.

Per a que l'observador faci la seva feina, cal determinar el valor de la seva matriu L. Una manera de trobar aquest valor, que és la que s'empra en aquest projecte, consisteix en fer un problema d'assignació de pols, molt semblant al realitzat per al càlcul del controlador fet en l'apartat 3.4.2, encara que amb una diferència [3]. L'equació resultant de l'observador d'estat és:

$$\dot{\hat{x}}(t) = (A - LC)\hat{x}(t) + Bu(t)$$

Aquesta és molt similar a l'equació del controlador, tret que la matriu incògnita (L en aquest cas) no està en la mateixa posició. No obstant, es pot fer ús d'una propietat de les matrius, la qual diu que els valors propis d'una matriu i els de la seva transposada són els mateixos:

$$eig(A - LC) = eig(A - LC)^T = eig(A^T - C^T L^T)$$

La transformació de l'últim pas permet fer un problema d'assignació de pols amb les matrius A^T i C^T, per obtenir L^T i la comana `place` que es fa servir per al càlcul del controlador (en l'apartat 3.4.2). Llavors només resta transposar aquest resultat per, ara sí, disposar de la matriu de l'observador d'estat.

L'objectiu d'un observador és que sigui ràpid. Molt més ràpid que el sistema original, de fet. Com que és un mecanisme que queda implementat (en aquest projecte) a nivell de software, es pot dissenyar aquest sense risc a trobar saturacions o que alguns valors interns es tornin molt elevats en algun punt.

S'aplica, llavors, la regla general de que l'observador d'estat sigui deu vegades més ràpid que el sistema original, a fi que en pugui fer-ne un bon seguiment. No obstant, com després de la modificació inicial el sistema $ssHM$ té un pol a 0, s'ha multiplicat per 10 el seu altre pol (-5.82) i s'ha imposat un altre a -5.83, que també queda augmentat pel mateix factor.

Les matrius de l'observador d'estat resultants per a cada subsistema són:

$$L_{Roll} = \begin{pmatrix} 507{,}2480 \\ -766{,}9483 \end{pmatrix}, \qquad L_{Roll2V} = 1{,}7328$$

$$L_{Pitch} = \begin{pmatrix} 243{,}2364 \\ -443{,}7702 \end{pmatrix}, \qquad L_{Pitch2U} = -1{,}9451$$

$$L_{Yaw} = 0{,}0418, \qquad L_{HM} = \begin{pmatrix} 1289{,}2 \\ 4{,}8 \end{pmatrix}$$

3.4.4 Creació dels adequadors de consigna

Com a darrer pas previ a executar la simulació, és necessari fer el càlcul de l'última part del sistema de control, els adequadors de consigna. Si les sortides del controlador s'intenten comparar amb les consignes introduïdes, es troba el problema que les unitats no coincideixen. Això es deu a que les consignes tenen les unitats de les sortides del sistema (ja que són les variables visibles i les que volem controlar), mentre que les sortides del controlador tenen les unitats de les variables d'entrada al sistema.

A causa d'això, els senyals no es poden combinar, ja que, a més de les unitats, treballen també amb escales diferents. Per això es fa servir l'adequador de consigna. Aquest bloc adapta les senyals de la consigna, i les amplifica o redueix per ajustar-les als valors corresponents de les entrades al sistema.

Per a realitzar el càlcul de l'adequador, s'empra la següent fórmula, que fa servir les diferents matrius de cada subsistema així com la del seu controlador de realimentació corresponent segons [3]:

$$adeq = \left(\frac{-C}{(A - BK) * B}\right)^{-1}$$

El resultat per a cada component de la consigna és el següent:

$$adeq_{Roll} = 14{,}4016$$

$$adeq_{Roll2V} = 0{,}4190$$

$$adeq_{Pitch} = 9{,}7972$$

$$adeq_{Pitch2U} = -0{,}3250$$

$$adeq_{Yaw} = 1{,}5807$$

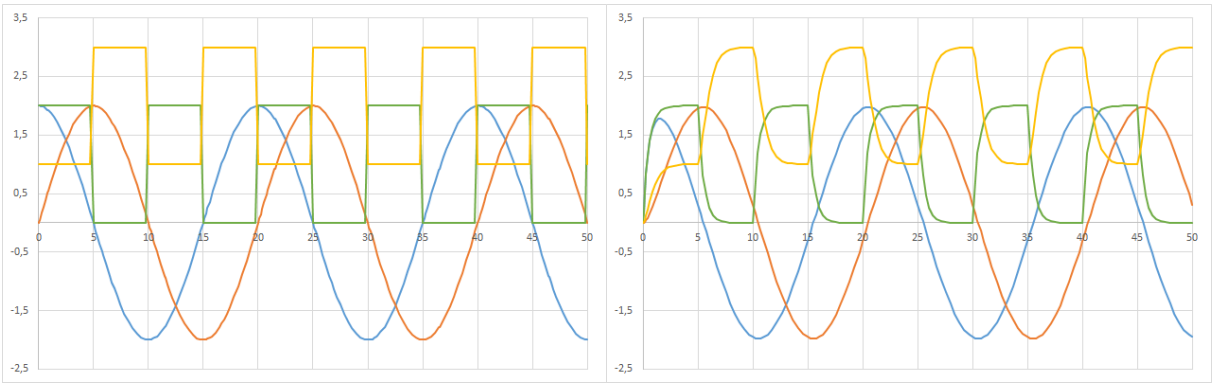
$$adeq_{HM} = 0{,}7801$$

3.4.5 Prova de simulació de la realimentació d'estat

Finalment, el model està llest per ser controlat amb el realimentador d'estat. Recordant el que s'exposa en l'apartat 3.2, primer es fa la prova de simulació en el model completament lineal, per fer una validació inicial del model. Després, un cop feta, s'introdueix el mateix controlador al model complet, per fer el control amb les no-linealitats.

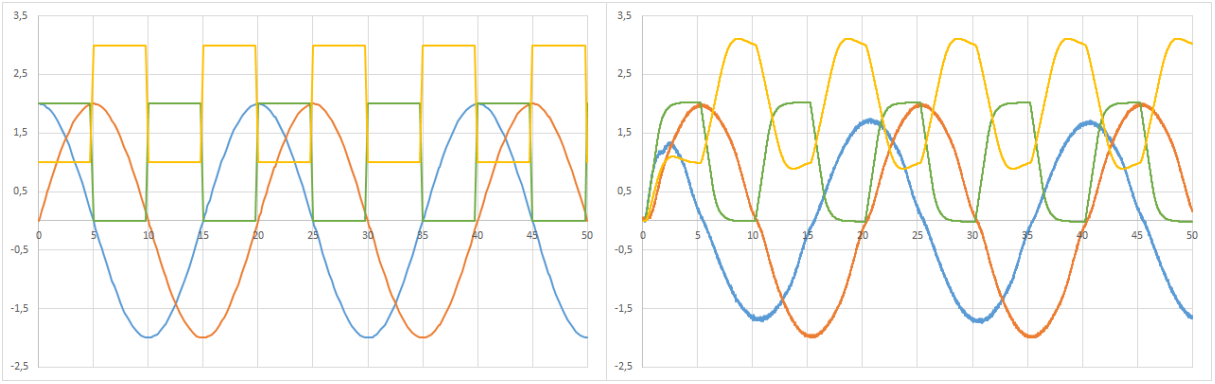
Cal dir que a l'hora d'executar la simulació, tot i que se'ls ha calculat l'observador i el controlador, les senyals que haurien d'anar cap als sistemes $ssRoll$ i $ssPitch$ estan tallades, ja que aquests subsistemes no afecten de forma directa a l'evolució de les variables d'estat que es pretén controlar (velocitats, *Yaw* i altura).

En aplicar la consigna de prova al model lineal s'obté el següent resultat a les sortides:



Figs. 3.5 i 3.10: Consigna de prova aplicada al model lineal i la seva resposta. Les velocitats lateral i frontal estan representades en blau i taronja respectivament, el Yaw *en verd i l'altura en groc. Mesures en metres per segon, radians i metres, eix horitzontal en segons.*

Fent l'anàlisi de la resposta, el seguiment que el sistema fa de les consignes sembla prou bo. S'aprecia un lleuger retard en les dues velocitats (molt lleu), mentre que la resposta del *Yaw* i l'altura als graons programats és l'esperada, arribant al valor final després d'un temps d'establiment. Tot seguit, es porta el mateix controlador al model complet, i se n'observa la resposta:



Figs. 3.5 i 3.11: Consigna de prova aplicada al model complet i la seva resposta. Les velocitats lateral i frontal estan representades en blau i taronja respectivament, el Yaw *en verd i l'altura en groc. Mesures en metres per segon, radians i metres, eix horitzontal en segons.*

Ara sí que hi ha una diferència més notable respecte als valors esperats. La velocitat lateral no acaba d'arribar al valor esperat. Tot i que la diferència és poca, s'aprecia la menor amplitud al costat de la velocitat frontal, que sí arriba al valor desitjat de 2.

La primera correcció que s'intenta fer és la d'augmentar el pol corresponent lleugerament, per veure si es presenta una millora. Per tant, el pol desitjat del subsistema $ssRoll2V$ es puja a un valor de -3, es recalculen els adequadors i es torna a repetir la simulació:

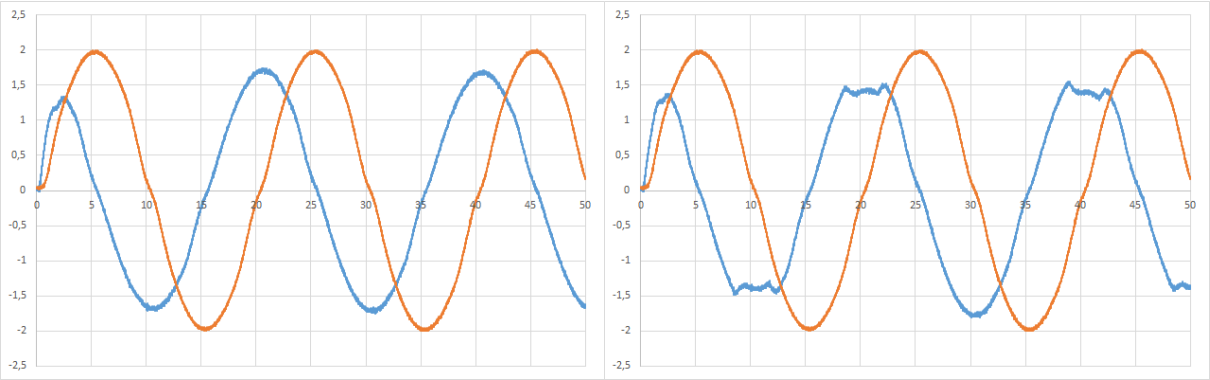


Fig. 3.12 i 3.13: Comparació entre la velocitat lateral (blau) i la frontal (vermell), abans i després d'augmentar-li el pol a -3. Mesures en metres per segon, eix horitzontal en segons.

Després de fer el pol més ràpid, el sistema reacciona d'aquesta manera, tallant la corba abruptament, i el valor del pic no millora (de fet baixa per sota de $1{,}5$). Tampoc millora la situació ajustar-lo a valors més petits, com $1{,}5$ ó 1, ja que aquests fan el sistema més lent i aleshores no té temps d'arribar al valor desitjat tampoc. Per tant, tot i la petita pèrdua d'amplitud, es dóna com a vàlid el pol a -2.

3.5 Control en llaç tancat per controlador PID

L'altra manera de realitzar el control en llaç tancat d'un sistema que es tracta en el projecte, a part de la realimentació d'estat que s'ha fet servir anteriorment en l'apartat 3.4, és la realimentació de l'error amb un controlador PID.

Aquest tipus de controlador pren el valor de les sortides i el resta al de les consignes directament, entenent aquest resultat com a l'error del sistema. Aleshores, actua sobre aquest senyal d'error de diferents maneres, ja que està compost per tres parts diferenciades [2]:

$$L(s) = K_p + \frac{K_i}{s} + K_d s$$

- La part proporcional (K_p) corregeix el valor present de l'error.
- La part integral (K_i) actua sobre els valors passats de l'error, acumulant-los mitjançant la seva integral. D'aquesta manera, és capaç de corregir fonts d'error contínues.
- La part derivativa (K_d) actua sobre els possibles valors futurs de l'error, prenent la seva derivada (tendència) actual.

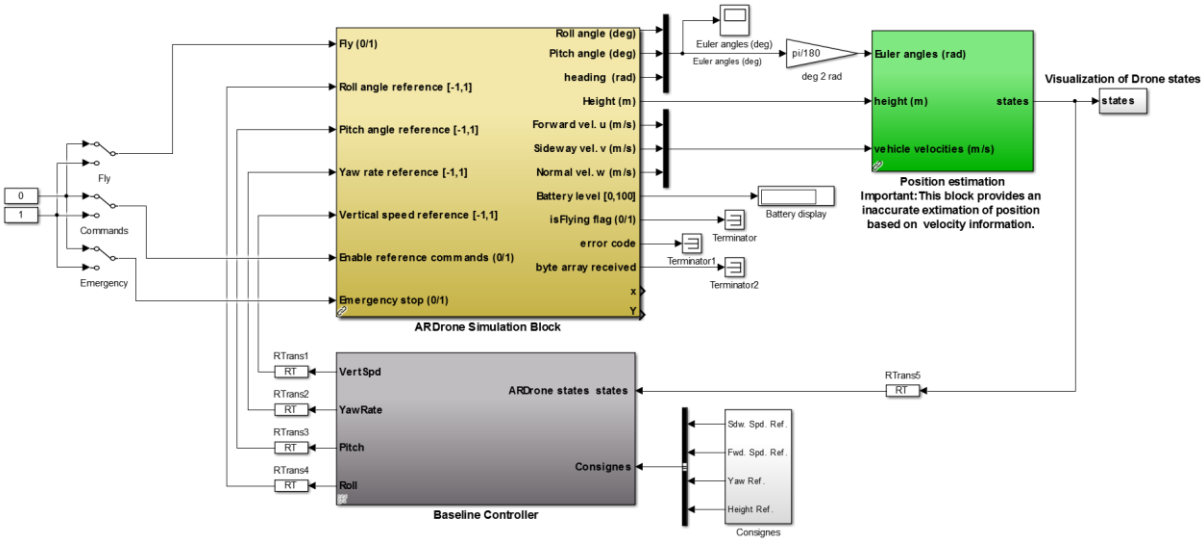


Fig. 3.14: Model del sistema en llaç tancat, amb un controlador PID (bloc gris).

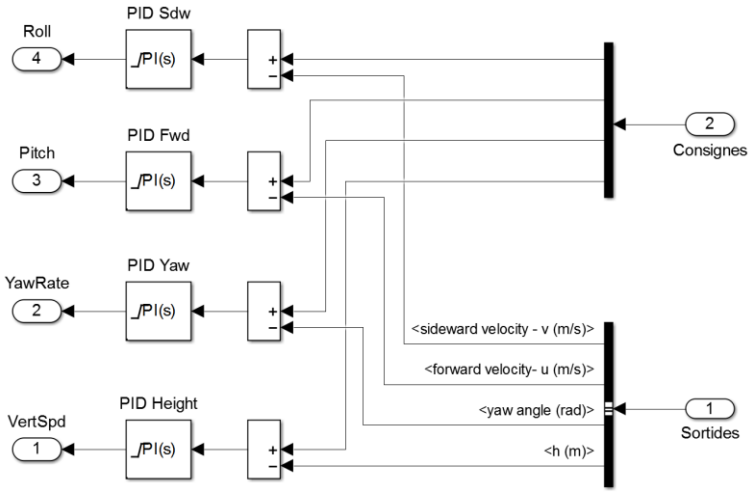


Fig. 3.15: Estructura interna del controlador PID (bloc gris de la Fig. 3.14). S'hi poden veure els controladors per a cada subsistema del model.

El controlador PID presenta una clara diferència respecte al realimentador d'estat, per exemple, en que no necessita un observador d'estat, al no caldre conèixer les variables d'estat internes. A més, com compara les sortides del sistema amb les diferents consignes directament, i no necessita un adequador que en faci un ajust. Això fa que tant el càlcul com l'aplicació d'aquest controlador siguin més senzills.

La implementació d'aquest mecanisme de control en el SimuLink també és molt senzilla, ja que aquest disposa d'un bloc específic de controlador PID. Dintre d'aquest, es poden configurar els diferents paràmetres que en modifiquen el seu funcionament, que són les tres constants anteriorment mencionades: K_p, K_i i K_d.

3.5.1 Càlcul i ajust del controlador PID

En primer lloc, cal recordar que primer es treballa amb el model completament lineal per a, després, aplicar el controlador trobat inicialment al model complet, segons s'explica a l'apartat 3.2.

Aquest procés de càlcul consta de dues fases. En la primera, es fa servir una opció dins del propi bloc de cada PID, l'anomenada "Tune", la qual després de seleccionar quines parts del controlador es vol que estiguin actives (P, I o D) carrega un mòdul que realitza un primer ajust dels paràmetres del bloc, permetent modificar-los mitjançant una sèrie de dials. Després, com que els valors que aquest sistema selecciona no són números fàcils de treballar, posteriorment aquests valors s'arrodoniran a uns altres que funcionin correctament i siguin més senzills de tractar.

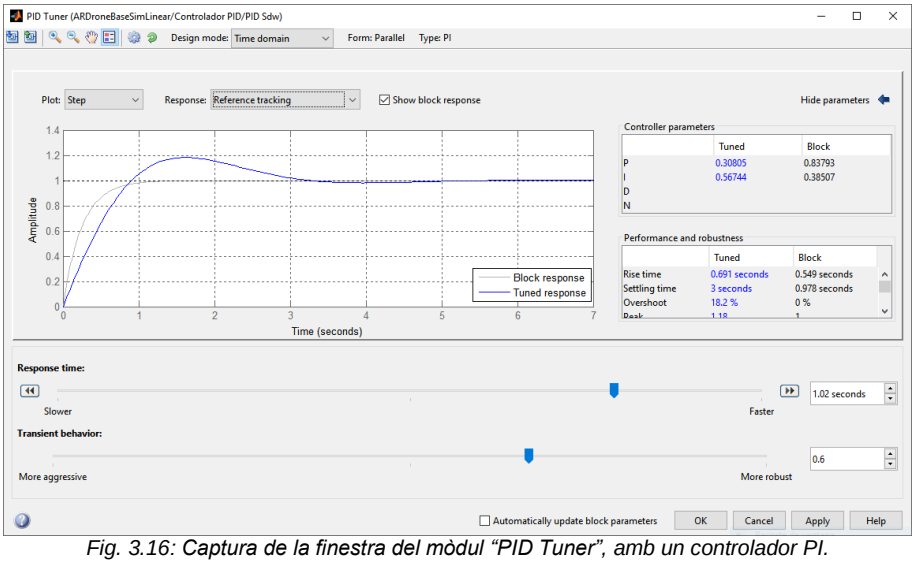


Fig. 3.16: Captura de la finestra del mòdul "PID Tuner", amb un controlador PI.

En aquesta finestra del "PID Tuner" es pot veure l'aspecte que té, i quines parts el conformen. Els dos controls lliscants permeten alterar el temps de resposta del controlador (superior) així com la brusquedat del seu comportament transitori (inferior). Al mateix temps, es representa per als valors seleccionats la resposta del sistema controlat a un graó unitari, així com la resposta al mateix graó del sistema sense controlador, de forma comparativa. A més, a la part dreta es poden fer aparèixer els valors que s'estan configurant al controlador, que s'actualitzen en temps real.

Provant diferents configuracions, el criteri que s'ha fet servir per a aquests valors inicials és molta robustesa i un temps de resposta aproximat d'un segon. D'aquesta forma, s'ajusten tots els controladors per a cada subsistema. De la mateixa manera que per al realimentador d'estat, els subsistemes $ssRoll$ i $ssPitch$ es deixen de banda, ja que no afecten a les variables que es volen controlar (velocitats, *Yaw* i altura), i aquests no disposen del seu PID. Els valors inicials dels controladors en qüestió són:

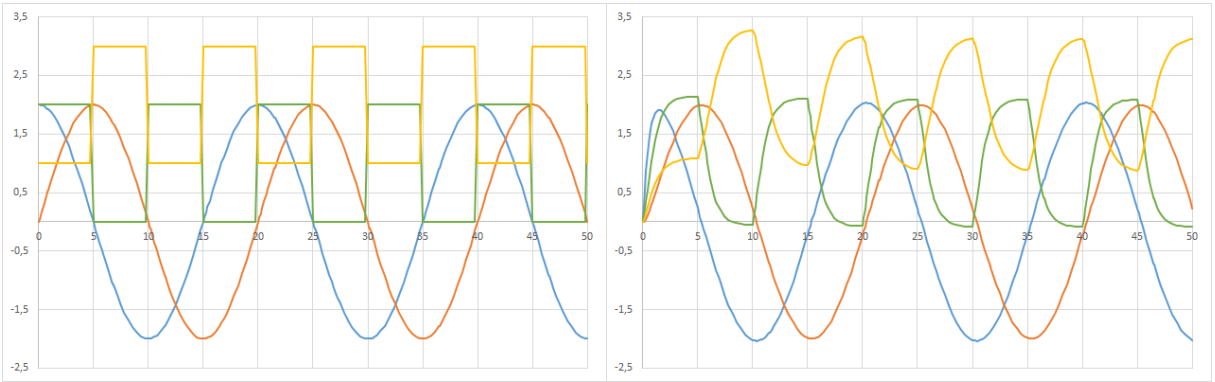
$$PI_{ssRoll2V}: \begin{cases} K_p = 0{,}8206 \\ K_i = 0{,}3771 \end{cases}, \qquad PI_{ssPitch2U}: \begin{cases} K_p = -0{,}6499 \\ K_i = -0{,}4322 \end{cases}$$

$$PI_{ssYaw}: \begin{cases} K_p = 3{,}1609 \\ K_i = 0{,}2213 \end{cases}, \qquad PI_{ssHM}: \begin{cases} K_p = 5{,}4430 \\ K_i = 0{,}3810 \end{cases}$$

Finalment s'opta per controladors PI, sense part derivativa. Això respon a diversos motius, que depenen del funcionament de cada part del controlador quan es comporta de forma transitòria. La part proporcional és la que treballa inicialment, mentre la part integradora encara no ha tingut temps d'acumular error suficient per ser significativa. Un cop s'acosta més la resposta al valor final, els seus papers s'inverteixen, tornant-se quasi irrellevant la proporcional i prenent protagonisme la integradora. També aporta els avantatges de neutralitzar el soroll (que té un valor mig de 0) i corregir un possible error permanent, si n'hi hagués.

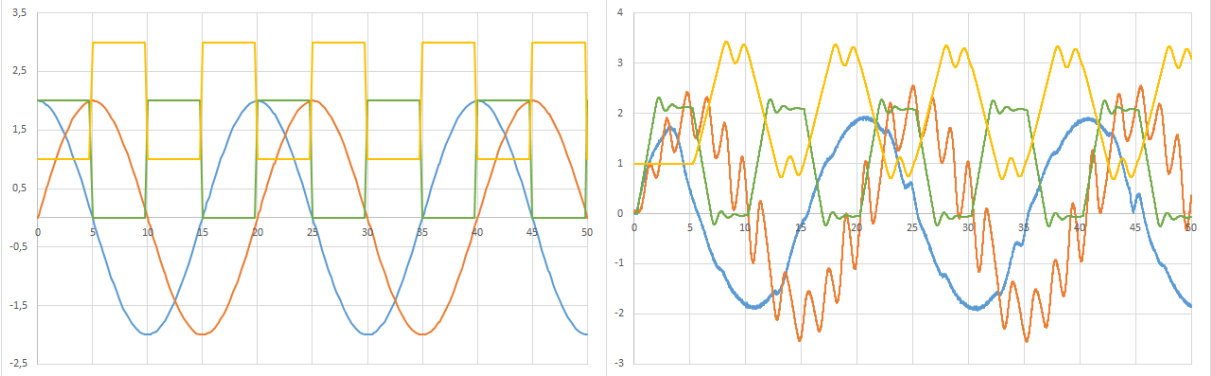
No obstant, encara que la part derivativa habitualment potencia el control dels sistemes, en aquest cas no és d'ajuda, ja que el sistema que es vol controlar presenta un soroll molt marcat un cop no es treballi amb el model lineal. Davant de sorolls, aquesta es dispara erràticament, derivant-lo juntament amb l'error que es vol controlar. D'aquesta manera, es justifica la seva omissió en el controlador final.

Finalment, s'executa el model lineal, aplicant la consigna de prova:



Figs. 3.5 i 3.17: Consigna de prova aplicada al model lineal i la seva resposta. Les velocitats lateral i frontal estan representades en blau i taronja respectivament, el Yaw *en verd i l'altura en groc. Mesures en metres per segon, radians i metres, eix horitzontal en segons.*

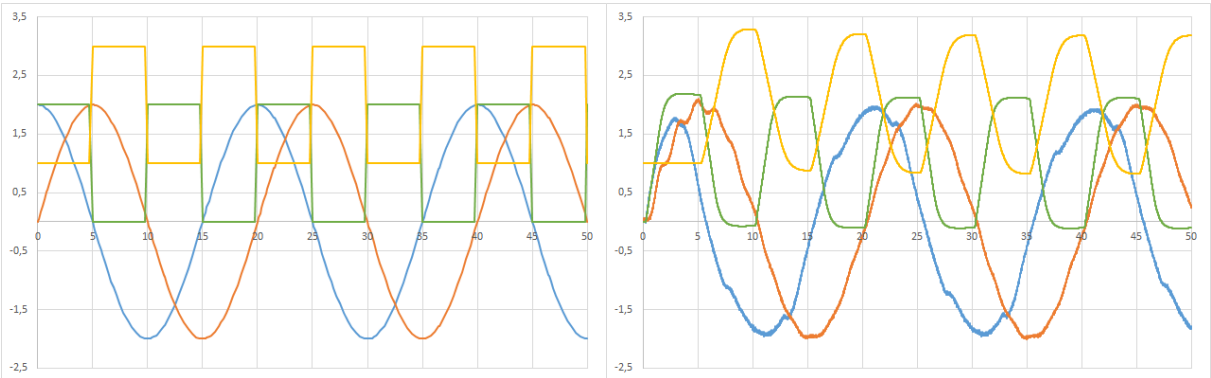
Els resultats semblen bastant correctes amb aquest controlador. Totes les senyals s'acosten bastant a les consignes, tenint en compte els retards i el moviment suau de les respostes als graons. Per tant, s'introdueix el controlador tal com està al model complet i s'executa:



Figs. 3.5 i 3.18: Consigna de prova aplicada al model complet i la seva resposta. Les velocitats lateral i frontal estan representades en blau i taronja respectivament, el Yaw *en verd i l'altura en groc. Mesures en metres per segon, radians i metres, eix horitzontal en segons.*

El comportament en aquest model és molt erràtic, totes les corbes presenten oscil·lacions, molt pronunciades en el cas de la velocitat frontal. Donat que el comportament no és el desitjat (tot i que és previsible que no ho sigui, ja que ara intervenen les no-linealitats), es procedeix a retocar el valor dels components del PID perquè la resposta s'aproximi tot el possible a les consignes d'entrada. També es busca que els valors d'aquest no siguin nombres massa extensos, a fi que sigui fàcil reproduir els resultats posteriorment.

Després de retocar els diferents paràmetres dels controladors, s'arriba a aquest resultat:



Figs. 3.5 i 3.19: Consigna de prova aplicada al model complet i la seva resposta. Les velocitats lateral i frontal estan representades en blau i taronja respectivament, el Yaw *en verd i l'altura en groc. Mesures en metres per segon, radians i metres, eix horitzontal en segons.*

Segueixen apareixent unes petites oscil·lacions durant el control de les velocitats, però són molt lleus i la forma general de les ones és bona, així com la seva amplitud. El control del *Yaw* i l'altura ara funciona correctament. Els valors dels controladors que generen aquesta resposta són:

$$PI_{ssRoll2V}\begin{cases}K_p = 0{,}4\\ K_i = 0{,}3\end{cases}$$

$$PI_{ssPitch2U}\begin{cases}K_p = -0{,}4\\ K_i = -0{,}3\end{cases}$$

$$PI_{ssYaw}\begin{cases}K_p = 1\\ K_i = 0{,}1\end{cases}$$

$$PI_{ssHM}\begin{cases}K_p = 1\\ K_i = 0{,}1\end{cases}$$

els quals es donen finalment com a vàlids per a introduir-los en el dron real en el següent capítol.

4 IMPLEMENTACIÓ EN EL DRON REAL

Finalitzat el primer objectiu, que és el disseny d'un controlador fent servir el simulador, ara es passa al segon objectiu, que és el d'agafar aquest mateix mecanisme de control i validar-lo en un dron real. En aquest, es fan una sèrie d'experiments, amb la finalitat d'acabar d'ajustar els valors del controlador a uns més correctes i que responguin millor a les consignes que s'introdueixin al sistema.

El model de SimuLink que cal fer servir ara és una mica diferent. El seu aspecte extern és pràcticament el mateix (tret d'algunes petites diferències). La principal diferència, però és interna. Mentre que el bloc principal del model en la part de simulació (el de color groc a les figures 3.1, 3.6 i 3.14) s'encarrega de modelitzar el dron, en aquesta part el bloc té una funcionalitat completament diferent, que és enviar els valors de les entrades a l'aparell i retornar la lectura dels sensors.

No obstant, de cara a treballar amb el controlador, les entrades i sortides del bloc sí que són exactament iguals, i a efectes pràctics es poden prendre com a sistemes equivalents. Això comporta que no cal canviar per a res l'estructura del sistema de control, només ajustar-ne els paràmetres.

L'estructura del model queda, aleshores, de la següent manera (molt similar als models anteriors en quant a representació externa):

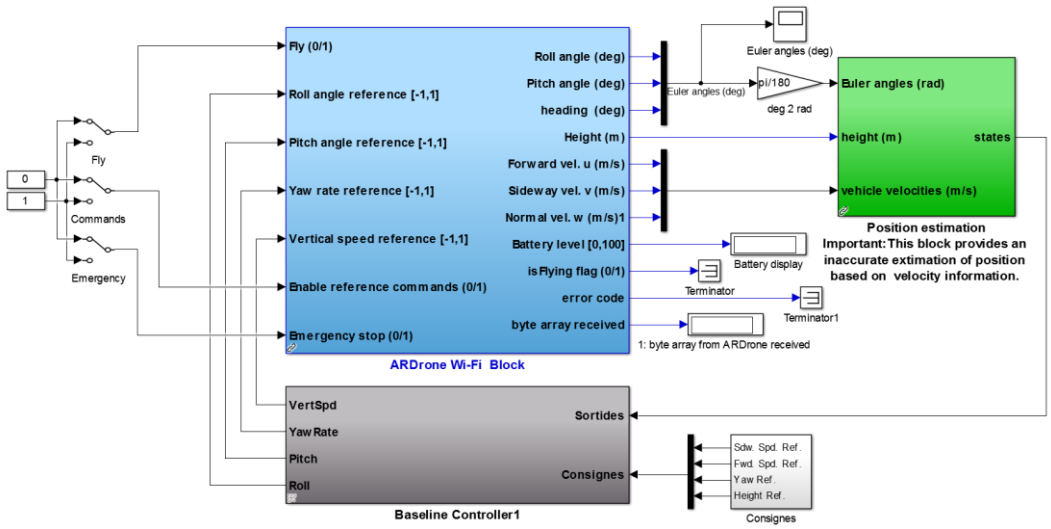


Fig. 4.1: Model del sistema en llaç tancat amb controlador PID, en la versió amb connexió externa al dron.

Malauradament, no ha estat possible fer l'experimentació del controlador per realimentació d'estat per problemes tècnics, així que tota la segona fase es treballa únicament amb el controlador PID.

En la fase de simulació, es va estar treballant principalment amb el realimentador d'estat, el qual no va funcionar correctament. Contínuament donava errors, presentava gràfiques que evolucionaven de forma estranya, i apareixien inestabilitats que tenien a infinit. Davant aquesta situació, i per a poder començar a realitzar l'experimentació, es va decidir començar a treballar també amb un controlador PID, que és el que s'ha portat finalment a la fase d'experimentació.

En l'última setmana abans de fer l'entrega, però, es va restaurar una còpia de seguretat del sistema amb realimentador d'estat, a fi de fer algunes captures de pantalla de l'estructura i el procediment. Es va provar a executar una simulació, i en aquest cas el model funcionava correctament. No obstant, ja era massa tard per a poder experimentar amb l'aparell a temps per a l'entrega de la memòria, de forma que es va decidir no desenvolupar aquesta part.

Tot i així, el que sí que s'ha fet és realitzar la part de simulació del realimentador d'estat, ja que ara funciona correctament i dóna bons resultats.

4.1 Implementació del controlador PID

Un cop feta la substitució del controlador per defecte que portava el model per el PID (i havent fet les connexions i afegit els blocs necessaris per a analitzar les dades), s'ha procedit a realitzar l'experiment. No obstant, com que no es coneix la capacitat del dron de seguir la trajectòria desitjada amb aquest controlador acabat de sortir de la simulació, primer s'han fet uns experiments manuals, a fi de fer un ajust si fos necessari.

En aquests, els controladors per a cada subsistema s'han validat un a un, fent que les consignes en tots ells siguin constants, mentre que al que en aquell moment s'està avaluant se li introdueix una consigna que es controla manualment. Això s'aconsegueix connectant-hi un parell de blocs de valor constant, i alternant-los amb un interruptor.

Quan tots els controladors han estat validats, aleshores s'ha procedit a treballar amb la consigna preparada anteriorment. Tot i això, s'ha fet un canvi per evitar un descontrol del dron. Com que les funcions que s'introdueixen a les velocitats tenen una amplitud fixa (sinus i cosinus), alguna d'elles no estarà a zero quan es comenci l'experiment. Això provocarà una correcció massa gran des de l'inici, que pot donar problemes.

Per solucionar-ho, es fa que ambdues consignes estiguin multiplicades per la sortida d'un sistema de primer ordre, al qual se li aplica un graó unitari en començar l'experiment. D'aquesta manera, l'amplitud de les dues va augmentant progressivament fins a estabilitzar-se en el valor original passats uns segons.

4.1.1 Control de l'altura

Per al control de l'altura, la validació inicial s'ha fet en primerament amb el sistema de blocs constants-interruptor descrit anteriorment, alternant entre uns valors de 1 i 2 metres. L'aspecte dels blocs de la consigna és aquest:

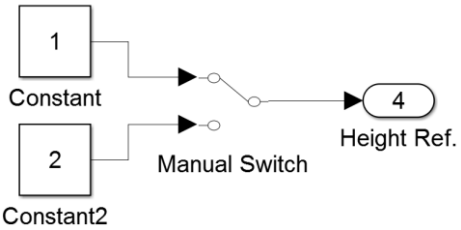


Fig. 4.2: Consigna manual de l'altura.

El resultat final d'aquest mecanisme és una sèrie de graons que canvien entre els dos valors, però que en comptes d'estar programats per temps es disparen manualment. Tot seguit, aquesta consigna s'aplica al dron i s'obté la següent resposta (amb uns paràmetres de $K_p = 1$ i $K_i = 0{,}1$):

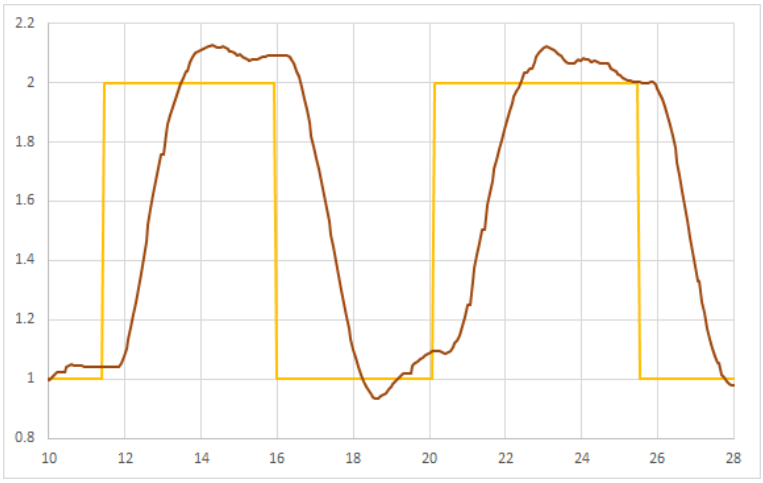
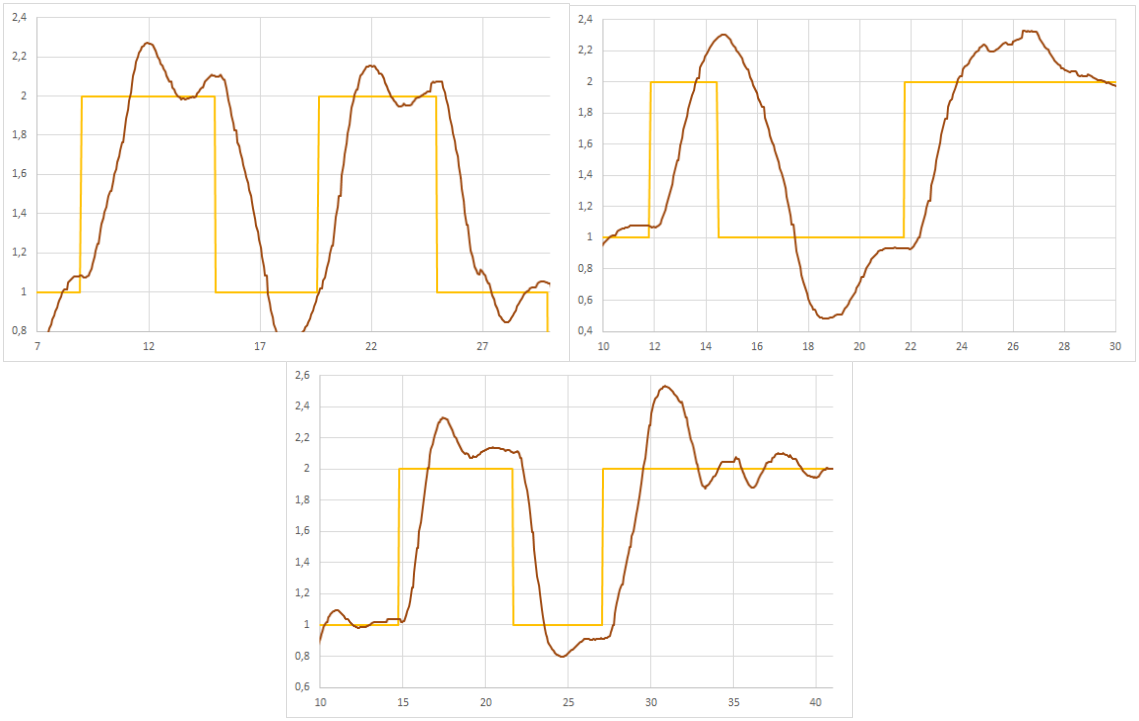


Fig. 4.3: Comparació entre la consigna d'altura i la lectura dels sensors del dron, amb el controlador kp=1, ki=0.1. Mesures en metres, eix horitzontal en segons.

En aquests resultats es pot apreciar un lleuger sobrepuig que es corregeix en pocs segons, que amb aquesta consigna té un valor aproximat de 0,1 metres, la qual cosa no és un problema. Encara que aquest controlador funciona prou bé, és convenient fer algunes proves per veure si es pot millorar.

A les gràfiques següents estan representats diferents canvis. En el primer cas, s'ha augmentat la K_p a $1{,}5$, en el segon s'ha augmentat la K_i a $0{,}3$ i, finalment, en el tercer s'han augmentat totes dues alhora. Els resultats són:



Figs.4.4, 4.5 i 4.6 : Comparació entre la consigna d'altura i la lectura dels sensors del dron, aplicant diferents modificacions al controlador PID. Mesures en metres, eix horitzontal en segons.

Tal com s'aprecia en les gràfiques, després d'augmentar la part proporcional, apareix un sobrepic molt més pronunciat (aproximadament d'un 25%), igual que en augmentar la part integral. Quan es modifiquen les dues alhora, a més, es genera una oscil·lació bastant marcada (tot i que passat el temps remiteix). Per tant, en el cas de l'altura, es pren la decisió de fer servir el controlador inicial sense modificar ($K_p = 1$ i $K_i = 0{,}1$).

4.1.2 Control del Yaw

El procés per validar el Yaw és molt similar al de l'altura, però presenta alguns canvis. La consigna manual s'ha fet amb una altra estructura, per poder-la posar en valor zero i valors negatius. El primer interruptor pot alternar entre uns valors de 0,5 i -0,5 rad, mentre que el segon interruptor alterna entre el primer i una constant de valor 0 rad. D'aquesta manera, la consigna final consisteix en graons de tres possibles valors: 0,5, 0 i -0,5 rad:

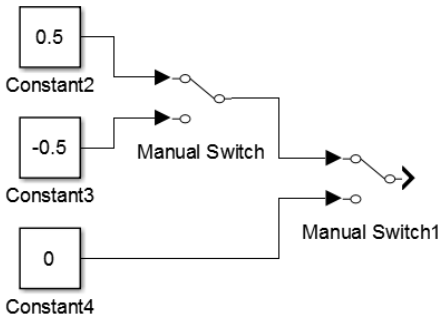


Fig. 4.7: Consigna manual del Yaw.

En aplicar aquesta consigna al dron s'obtenen els resultats següents:

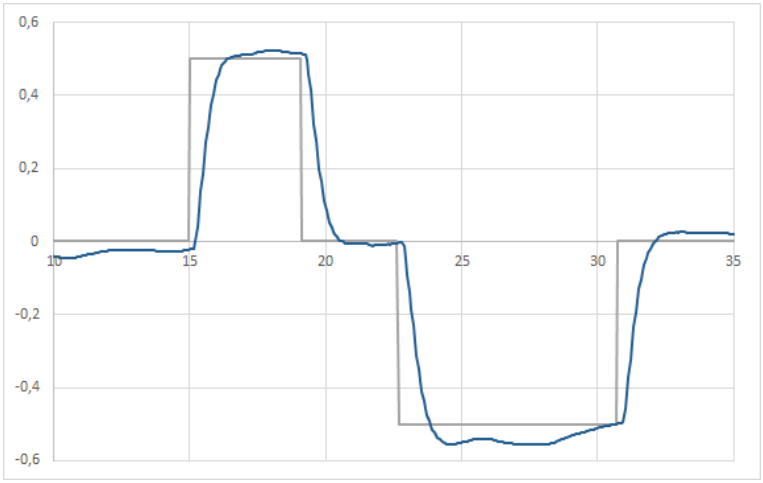


Fig. 4.8: Comparació entre la consigna del Yaw *i la lectura dels sensors del dron, amb el controlador kp=1, ki=0.1. Mesures en radians, eix horitzontal en segons.*

Dels resultats obtinguts es pot veure que el controlador que s'està avaluant és prou bo, arribant al seu valor final en un temps d'aproximadament un segon, sense que hi hagi un sobrepic marcat. Tot i això, s'ha fet una altra prova, modificant els valors del controlador a uns una mica més elevats. S'ha augmentat la seva K_p a 1.2 i la K_i a 0.3, per veure el seu comportament, que es mostra en la següent gràfica:

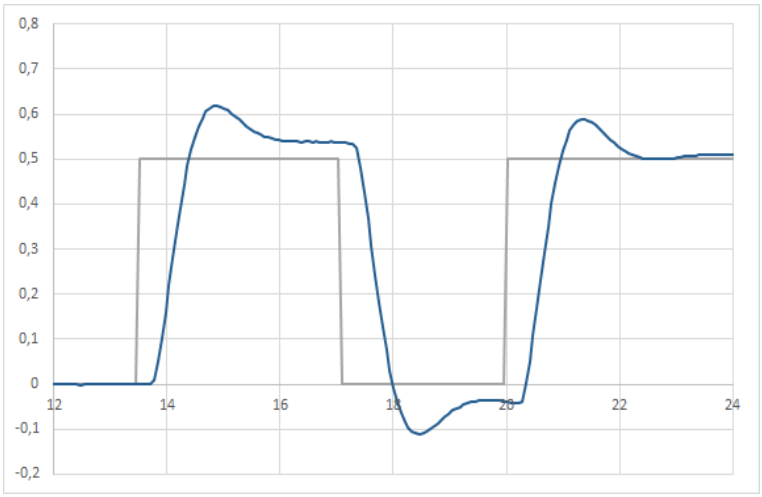


Fig. 4.9: Comparació entre la consigna del Yaw *i la lectura dels sensors del dron, amb el controlador kp=1.2, ki=0.3. Mesures en radians, eix horitzontal en segons.*

En augmentar a aquests valors comença a aparèixer un sobrepuig (també aproximadament del 20-25%), mentre que el sistema no es torna molt més ràpid en comparació. Per tant, també s'han escollit per a controlar el *Yaw* els valors inicials del PID de $K_p = 1$ i $K_i = 0{,}1$.

Cal deixar constància que en l'experimentació s'ha notat un problema amb el control del *Yaw*. Mentre que el valor d'aquest es manté al nivell desitjat amb petites variacions (segons els sensors i la gràfica d'ells), en l'observació visual del dron aquest realment està girant (tot i que a un ritme molt lent). Encara que en espais curts de temps aquest error no és molt significatiu, en experiments més llargs sí s'ha notat que el dron estava girat respecte al seu "zero" de referència, sense que això quedés reflectit per les gràfiques.

4.1.3 Control de la velocitat frontal

En el moment de validar el controlador de la velocitat frontal, la consigna que s'ha fet servir és la mateixa que la del *Yaw*. La raó és que si el dron s'està movent cap endavant i immediatament s'inverteix la seva velocitat, el descontrol pot ser molt gran. Per tant, primer es posa l'aparell en moviment a 0,5 m/s, tot seguit s'atura i quan està parat aleshores es fa anar en sentit contrari, per després tornar-lo a aturar:

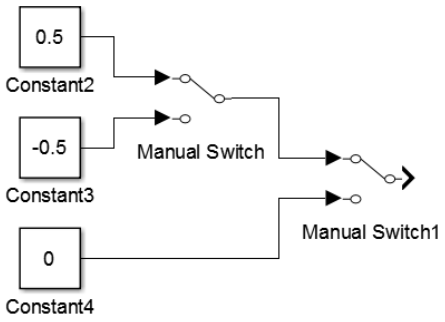


Fig. 4.10: Consigna manual de la velocitat frontal.

El resultat d'entrar aquesta consigna al dron genera aquesta resposta:

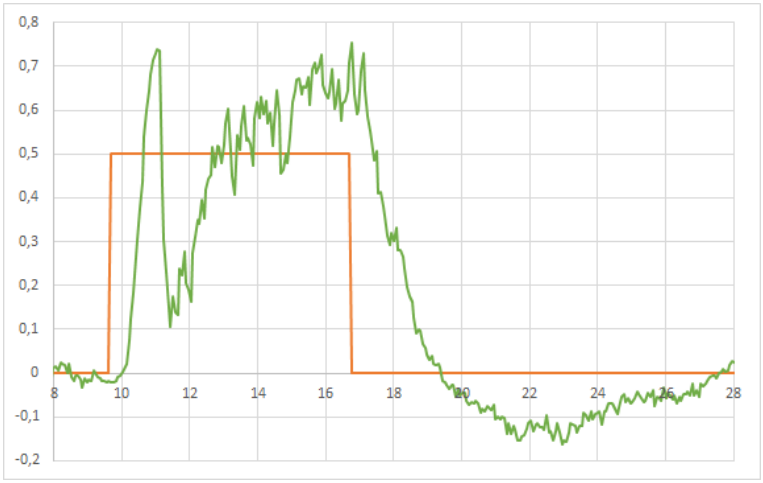


Fig. 4.11: Comparació entre la consigna de la velocitat frontal i la lectura dels sensors del dron, amb el controlador kp=-0.4, ki=-0.3. Mesures en metres per segon, eix horitzontal en segons.

En la gràfica es pot veure que el temps que triga el controlador a arribar al valor final és d'uns 3 segons, tot i que després el supera i ha de recuperar-se (és un sobrepuig molt llarg, ja que la part proporcional del controlador és de $-0{,}4$). Aleshores es pren la decisió d'augmentar-la, a fi que el controlador sigui més ràpid i la part integral no estigui tant de temps acumulant error. Per tant, s'ha duplicat la part proporcional del controlador, deixant la integral igual:

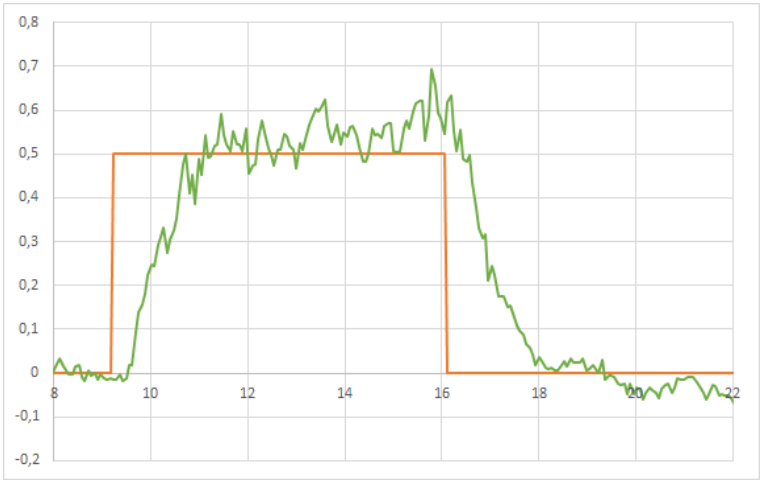


Fig. 4.12: Comparació entre la consigna de la velocitat frontal i la lectura dels sensors del dron, amb el controlador kp=-0.8, ki=-0.3. Mesures en metres per segon, eix horitzontal en segons.

En aquest experiment, el temps que triga el dron a arribar el seu valor final s'ha reduït (ara és d'uns 2 segons), i a més el sobrepuig present anteriorment ha desaparegut.

La millora del controlador és notable. No obstant, hi ha un soroll generat pel propi sensor, que tot i no ser significatiu a velocitats més grans, sí que s'aprecia molt a valors més baixos (com aquesta velocitat de 0,5 m/s). Per falta d'espai, no s'han pogut realitzar proves a velocitats majors, o sigui que aquest controlador modificat es pren com a vàlid, amb uns valors de $K_p = -0{,}8$ i $K_i = -0{,}3$.

4.1.4 Control de la velocitat lateral

El control de la velocitat lateral és molt similar a la frontal. Es fan servir els mateixos blocs en la consigna manual, amb els mateixos valors. El procés també consisteix en posar el dron en moviment, parar-lo, invertir el sentit i tornar-lo a parar. En aplicar la consigna de prova s'obtenen els següents resultats:

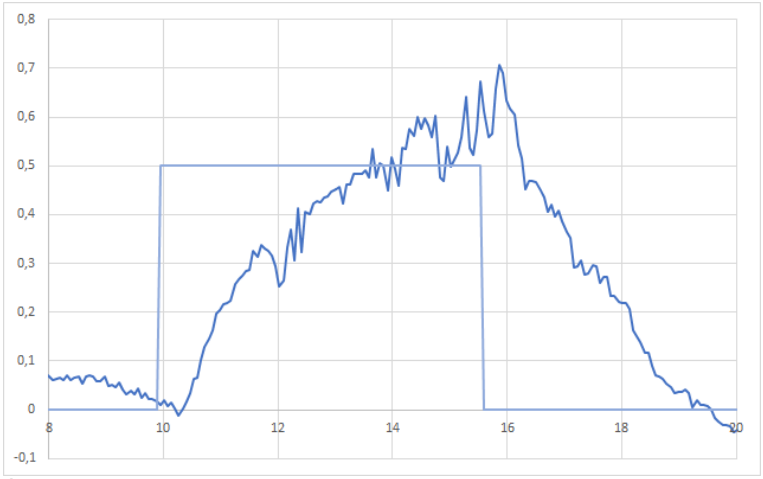


Fig. 4.13: Comparació entre la consigna de la velocitat lateral i la lectura dels sensors del dron, amb el controlador kp=0.4, ki=0.3. Mesures en metres per segon, eix horitzontal en segons.

De forma similar a l'experiment de la velocitat frontal, els valors s'han modificat per eliminar el sobrepuig amb el llarg temps d'establiment i fer el controlador més ràpid. Els valors que s'han provat són els mateixos, amb $K_p = 0{,}8$ i $K_i = 0{,}3$ (en aquest cas en positiu, ja que el subsistema no té un guany negatiu que calgui invertir):

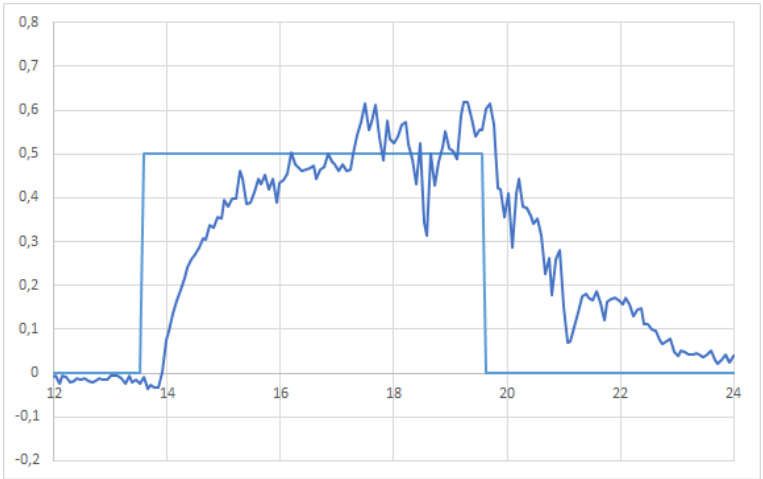


Fig. 4.14: Comparació entre la consigna de la velocitat lateral i la lectura dels sensors del dron, amb el controlador kp=0.8, ki=0.3. Mesures en metres per segon, eix horitzontal en segons.

De nou, la velocitat és prou bona i la millora és notable, per tant s'ha escollit finalment com a controlador de la velocitat lateral el de valors $K_p = 0{,}8$ i $K_i = 0{,}3$.

No obstant, cal fer un aclariment. De forma similar al que ocorre amb el *Yaw*, en el control de les velocitats s'aprecia un error que no queda reflectit a les gràfiques (sobretot quan el dron està estacionari). Mentre que els sensors donen un valor de 0 m/s (més el soroll i petites variacions) en l'observació de l'aparell es nota un moviment general en alguna direcció, un *drift* que té el dron i és innat a aquests tipus de sistemes.

4.1.5 Control complet del dron

Finalment, un cop validats tots els controladors per separat, cal aplicar-los tots alhora i veure com responen a una consigna més complicada, com la del vol en cercles. Per a fer una prova com aquesta, l'altura s'ha programat a un valor constant de 1 metre, mentre que el *Yaw* s'ha deixat a un valor de 0 rad, a fi d'eliminar diverses fonts d'error. En les velocitats lateral i frontal, s'han programat dues ones sinusoïdals, desfasades en $\pi/2$ (sinus i cosinus) perquè provoquin el moviment circular.

Per tant, l'aspecte final de les consignes és el següent:

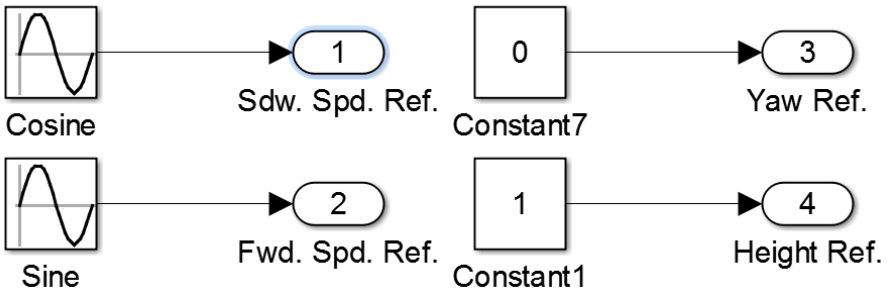
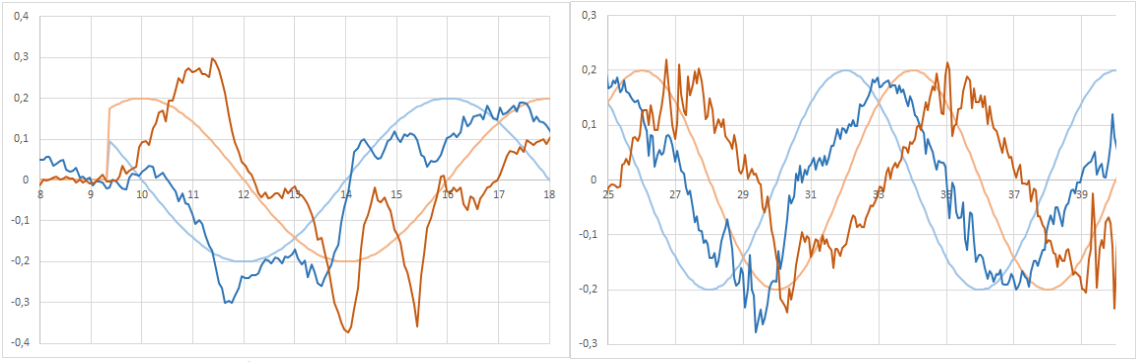


Fig. 4.15: Consignes per al vol en cercles.

I el resultat d'aplicar-la al dron és aquest:



Figs. 4.16 i 4.17: Comparació entre la consigna i la sortida en el vol en cercles, amb la velocitat lateral en taronja i la frontal en blau. La primera gràfica mostra la fase inicial, la segona la fase estable. Mesures en m/s, eix horitzontal en segons.

En la primera gràfica, quan la consigna s'introdueix de cop al sistema, l'error inicial respecte de les mesures fa que s'apliquin unes correccions molt marcades, la qual cosa fa que el sistema es comporti de forma estranya. Uns segons més tard, quan el dron ja s'ha estabilitzat en el seu moviment, el seguiment és adequat.

No obstant, cal corregir el problema de la fase inicial. Per això es fa una modificació a les consignes de les velocitats. Aquesta consisteix en multiplicar-les per uns valors constants, controlats per uns interruptors, els quals seran $0{,}5$ i 1. D'aquesta manera, el vol començarà amb uns cercles més lents, per després augmentar-ne la velocitat:

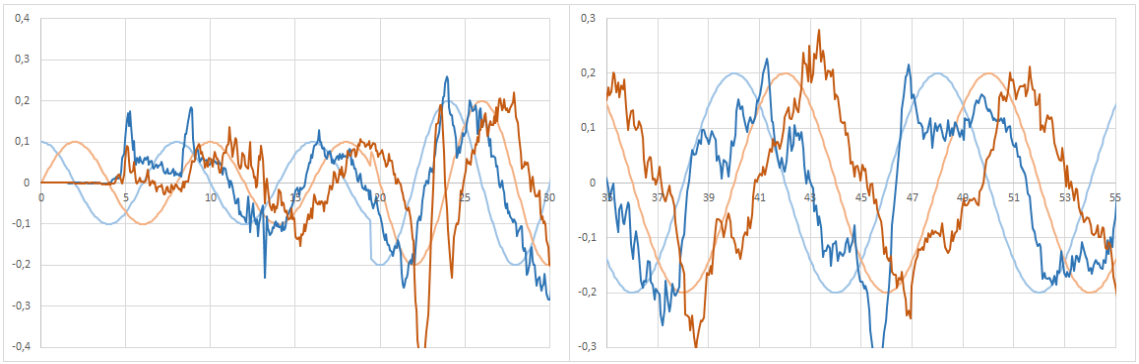


Fig. 4.18 i 4.19: Comparació entre la consigna i la sortida en el vol en cercles, amb la velocitat lateral en taronja i la frontal en blau. La primera gràfica mostra la fase inicial, la segona la fase estable. Mesures en m/s, eix horitzontal en segons.

Tot i fer aquesta modificació, el sistema segueix sent bastant irregular quan s'aplica la consigna. Fins i tot apareix un altre tram irregular quan s'amplifica de cop la consigna. Per tant, encara no està solucionat el problema. Per aconseguir-ho, aquestes funcions sinusoïdals han estat multiplicades per la sortida d'un sistema de primer ordre, el qual té com a entrada un graó unitari controlat manualment:

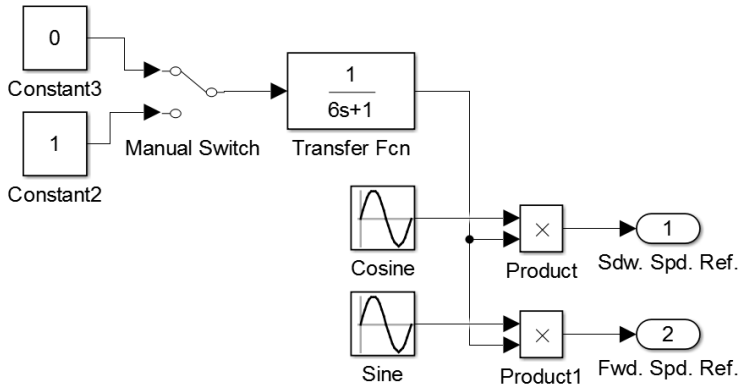


Fig. 4.20: Consigna final, amb el sistema de primer ordre.

D'aquesta manera, quan s'activi l'interruptor, el sistema anirà augmentant la seva sortida fins al valor final de 1 (guany unitari), trigant un temps d'una mica més de 6 segons en arribar-hi. Aquest valor multiplicarà l'amplitud de les dues ones, que aniran creixent progressivament fins a oscil·lar entre 0.2 i -0.2:

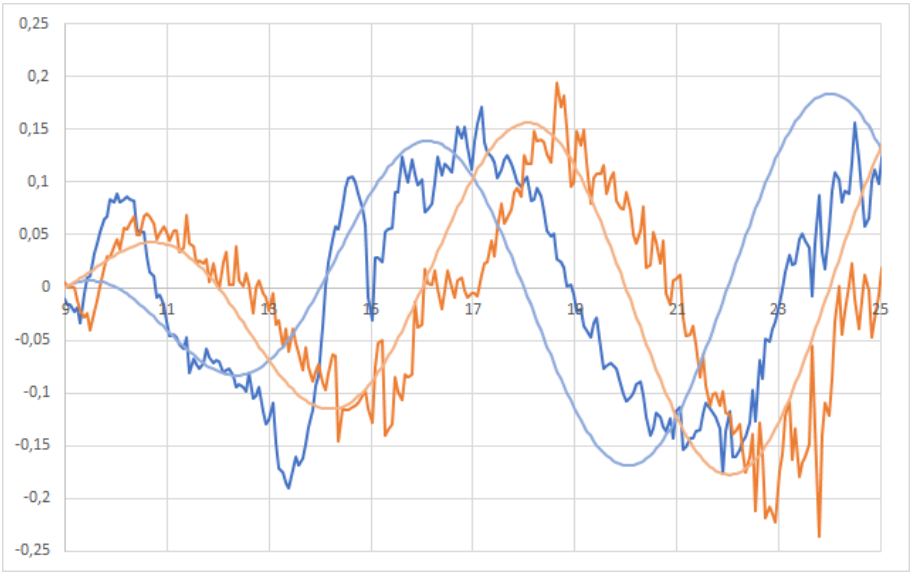


Fig. 4.21: Comparació entre la consigna i la sortida en el vol en cercles, amb la velocitat lateral en taronja i la frontal en blau. La primera gràfica mostra la fase inicial, la segona la fase estable. Mesures en m/s, eix horitzontal en segons.

Finalment, amb aquest creixement progressiu, s'aprecia que el dron no presenta grans irregularitats, i que fa un seguiment prou bo de la consigna, encara que cal comptar amb el retard inherent entre que la consigna s'envia, el dron respon i arriba la resposta.

Per tant el controlador PID que s'ha validat com a correcte és:

- Control de la velocitat frontal:

$$\begin{cases} K_p = -0.8 \\ K_i = -0.3 \end{cases}$$

- Control de la velocitat lateral:

$$\begin{cases} K_p = 0.8 \\ K_i = 0.3 \end{cases}$$

- Control del *Yaw*:

$$\begin{cases} K_p = 1 \\ K_i = 0.1 \end{cases}$$

- Control de l'altura:

$$\begin{cases} K_p = 1 \\ K_i = 0.1 \end{cases}$$

CONCLUSIONS/RECOMANACIONS

Les conclusions que es poden extreure d'aquest projecte són positives, però amb matisos que cal tenir en compte. Tot i així, tant en la part de simulació com, més important, en la part experimental, el resultat és satisfactori. El model de simulació i el dron real segueixen les consignes que se'ls imposen de forma bastant correcta. Se'n pot fer una anàlisi per separat:

- En el cas del model de simulació, el resultat és molt positiu. Tant en el cas del realimentador d'estat com en el del controlador PID la resposta és molt bona, ajustant-se de forma quasi perfecta a les corbes de velocitat, i reaccionant als graons en els dos altres subsistemes de forma àgil i sense excedir els valors imposats (sobrepuigs, *overshoots*), sense que s'apreciïn respostes erràtiques en cap dels casos.
- En el cas del dron real el resultat també és positiu. Tot i no ser tan bo com el de les simulacions, cal tenir en compte que en aquest hi intervenen molts altres factors, des d'imprecisions i aspectes presents en l'aparell que no estan modelitzats fins a pertorbacions externes, com corrents d'aire generades pel propi dron en volar a prop d'alguna paret, les quals fan que es comporti de forma erràtica quan s'hi acosta. Tot i així, tenint en compte tots aquests factors, el controlador PID estabilitza l'aparell de forma molt correcta, sempre que no es presentin pertorbacions molt fortes en el seu entorn.

No obstant, el principal problema que s'ha apreciat en aquest projecte (sobretot en la part d'experimentació), és la imprecisió dels sensors. L'altura ha estat l'excepció, ja que el sònar funciona correctament i mesura de forma directa aquesta variable. La resta, però, no són tan fiables.

Els sensors de velocitat generen molt soroll i, a més, són incapaços de detectar velocitats molt petites. Això provoca que el dron es mogui, sense que els sensors en siguin conscients. D'aquesta manera, un petit *drift* que l'afecti (corrents d'aire lleugeres) pot treure a l'aparell de la seva zona de treball d'aplicar-se aquest mecanisme de control en una situació real.

De manera similar, també es va apreciar que la mesura del *Yaw* no era coherent. Tot i prendre aquesta mitjançant un magnetòmetre (que hauria de determinar l'orientació absoluta en el pla respecte del Nord), sense que la mesura canviés de valor el dron clarament estava girant sobre sí mateix, la qual cosa és contradictòria amb les dades obtingudes.

Tenint en compte aquests problemes, la recomanació que faria de cara a una possible continuació del projecte seria la d'emprar algun sistema de posicionament extern, a fi de fer el control de l'aparell per posició i no per velocitats.

Les dues opcions més clares són: o bé implementant un sistema de marques òptiques al terra que l'aparell pot seguir per tal de saber a on està situat amb les seves càmeres; o bé es disposa d'un sistema de càmeres externes que determinin la posició i orientació del dron en l'espai. Això arreglaria qualsevol problema de *drift* (com el que pot aparèixer durant el control de velocitat) o d'orientació (quan el magnetòmetre no registra el gir de l'aparell).

PRESSUPOST, IMPACTE SOCIAL I AMBIENTAL

Pressupost del projecte

Concepte	Preu
Llicència d'estudiant de MatLab (inclou SimuLink)	69 €
Parrot AR.Drone Power Edition Inclou:	300 €
• AR.Drone 2.0	------
• 2 Bateries d'alta densitat	------
Temps invertit (orientativament, 540h, 10€/h)	5.400 €
Costos d'impressió i enquadernació	80 €
Total	5.930 €

Impacte social i ambiental

A nivell social, l'impacte que té aquest projecte es pot entendre com a les possibles lesions que pugui patir la gent si l'aparell es descontrolés, així com danys materials a l'entorn de treball si aquest xoca amb alguna cosa en el transcurs de les proves (llums, equips, etc.). De realitzar-se les proves en un espai obert, caldria tenir en compte tot un seguit de consideracions legals, ja que no està permès el vol d'aquests aparells en espais públics fora de zones dedicades a l'aeromodelisme.

En quant a l'impacte ambiental, apart de l'evident risc de lesions a fauna i flora (depenent d'on es faci volar l'aparell), també és preocupant és la contaminació acústica. Els drons quadricòpters d'aquestes dimensions tendeixen a emetre nivells elevats de soroll. Com a punt a favor, al igual que la gran majoria d'aparells similars, la seva alimentació és elèctrica, per tant no emet residus a l'atmosfera. Sí que s'ha notat, però, que quan es produeix algun impacte les hèlices poden rascar la carcassa de l'aparell, i desprendre fragments del PPE (polipropilè expandit) que la forma a l'entorn, el qual a la llarga poden fer malbé.

Quan el dron deixi de fer servei, llavors, caldrà disposar de tots els materials que el conformen de manera adequada. Entre els materials que en formen part, se'n poden destacar:

- Tota la circuiteria electrònica variada.
- Motors i cablejat.
- La carcassa de polipropilè (PPE).
- Els eixos principals de fibra de carboni.
- Bateries de polímer de liti (Li-Po).

Bibliografia

[1] KRAJNÍK, T; VONÁSEK, V; FISER, D; FAIGL, J. *AR-Drone as a Platform for Robotic Research and Education*. Czech Technical University in Prague.

[2] ÅSTRÖM, K. J. *PID Control*. From Control System Design, 2002.

[3] *OGATA, K. "Ingeniería de control moderna", 5ª Edición. Prentice Hall, 2010.*

Bibliografia complementària

PESTANA, J; SANCHEZ, J.L; MELLADO, I; CHANGHONG FU; CAMPOY, P. *AR Drone Identification and Navigation Control at CVG-UPM*. Joint Research Centre CSIC-UPM.

PRISTEAU, P.J; CALLOU, F; VISSIÈRE, D; PETIT, N. *The Navigation and Control technology inside the AR.Drone micro UAV*.

ÅSTRÖM, K. J. *Where to Place the Poles?*. Dept. of Automatic Control LTH, Lund University.

SOLER, M. *Estudi del control de trajectòria d'un UAV multirotor amb Robotic Operating System (ROS)*. Universitat Politècnica de Catalunya, 2015.

COSTAL, S. *Estudi d'algoritmes de prognosi aplicats a UAVs (Unmaned Aerial Vehicles) multirotors*. Universitat Politècnica de Catalunya, 2015.

ALCARAZ, C. *Estudi d'identificació d'un model matemàtic per a la realització del control de trajectòria d'un quadricòpter*. Universitat Politècnica de Catalunya, 2013.

REBULL, J. *Disseny i construcció d'un sistema volador no tripulat (UAV)*. Universitat Politècnica de Catalunya, 2014.

PÉREZ, J. *Diseño y construcción de un sistema volador no tripulado (UAV)*. Universitat Politècnica de Catalunya, 2014.

A Annex: Scripts de MatLab

A.1 Comprovació de controlabilitat i observabilitat

```
disp('-----------------------');
% SISTEMA ssRoll
wcRoll=ctrb(ssRoll.a,ssRoll.b);
rankwcRoll=rank(wcRoll);
if (rankwcRoll==2);
    disp('ssRoll es CONTROLABLE');
else
    disp('ssRoll no es CONTROLABLE');
end
woRoll=obsv(ssRoll.a,ssRoll.c);
rankwoRoll=rank(woRoll);
if (rankwoRoll==2);
    disp('ssRoll es OBSERVABLE');
else
    disp('ssRoll no es OBSERVABLE');
end
disp('-----------------------');
% SISTEMA ssRoll2V
wcRoll2V=ctrb(ssRoll2V.a,ssRoll2V.b);
rankwcRoll2V=rank(wcRoll2V);
if (rankwcRoll2V==1);
    disp('ssRoll2V es CONTROLABLE');
else
    disp('ssRoll2V no es CONTROLABLE');
end
woRoll2V=obsv(ssRoll2V.a,ssRoll2V.c);
rankwoRoll2V=rank(woRoll2V);
if (rankwoRoll2V==1);
    disp('ssRoll2V es OBSERVABLE');
else
    disp('ssRoll2V no es OBSERVABLE');
end
disp('-----------------------');
% SISTEMA ssPitch
wcPitch=ctrb(ssPitch.a,ssPitch.b);
rankwcPitch=rank(wcPitch);
if (rankwcPitch==2);
    disp('ssPitch es CONTROLABLE');
else
    disp('ssPitch no es CONTROLABLE');
end
woPitch=obsv(ssPitch.a,ssPitch.c);
rankwoPitch=rank(woPitch);
if (rankwoPitch==2);
    disp('ssPitch es OBSERVABLE');
else
    disp('ssPitch no es OBSERVABLE');
end
disp('-----------------------');
% SISTEMA ssPitch2U
```

```
wcPitch2U=ctrb(ssPitch2U.a,ssPitch2U.b);
rankwcPitch2U=rank(wcPitch2U);
if (rankwcPitch2U==1);
    disp('ssPitch2U es CONTROLABLE');
else
    disp('ssPitch2U no es CONTROLABLE');
end
woPitch2U=obsv(ssPitch2U.a,ssPitch2U.c);
rankwoPitch2U=rank(woPitch2U);
if (rankwoPitch2U==1);
    disp('ssPitch2U es OBSERVABLE');
else
    disp('ssPitch2U no es OBSERVABLE');
end
disp('-----------------------');
% SISTEMA ssYaw
wcYaw=ctrb(ssYaw.a,ssYaw.b);
rankwcYaw=rank(wcYaw);
if (rankwcYaw==1);
    disp('ssYaw es CONTROLABLE');
else
    disp('ssYaw no es CONTROLABLE');
end
woYaw=obsv(ssYaw.a,ssYaw.c);
rankwoYaw=rank(woYaw);
if (rankwoYaw==1);
    disp('ssYaw es OBSERVABLE');
else
    disp('ssYaw no es OBSERVABLE');
end
disp('-----------------------');
% SISTEMA ssH
wcH=ctrb(ssH.a,ssH.b);
rankwcH=rank(wcH);
if (rankwcH==2);
    disp('ssH es CONTROLABLE');
else
    disp('ssH no es CONTROLABLE');
end
woH=obsv(ssH.a,ssH.c);
rankwoH=rank(woH);
if (rankwoH==2);
    disp('ssH es OBSERVABLE');
else
    disp('ssH no es OBSERVABLE');
end
disp('-----------------------');
% SISTEMA ssHM
wcHM=ctrb(ssHM.a,ssHM.b);
rankwcHM=rank(wcHM);
if (rankwcHM==2);
    disp('ssHM es CONTROLABLE');
else
    disp('ssHM no es CONTROLABLE');
end
```

```
woHM=obsv(ssHM.a,ssHM.c);
rankwoHM=rank(woHM);
if (rankwoHM==2);
    disp('ssHM es OBSERVABLE');
else
    disp('ssHM no es OBSERVABLE');
end
disp('-----------------------');
```

A.2 Càlcul del controlador per realimentació d'estat

```
% Controlador ssRoll
eig1=eig(ssRoll.a);
VAPCont1=eig1*2;
K1=place(ssRoll.a,ssRoll.b,[VAPCont1]);

% Controlador ssRoll2V
eig2=eig(ssRoll2V.a);
VAPCont2=-2;
K2=place(ssRoll2V.a,ssRoll2V.b,[VAPCont2])

% Controlador ssPitch
eig3=eig(ssPitch.a);
VAPCont3=eig3*2;
K3=place(ssPitch.a,ssPitch.b,[VAPCont3]);

% Controlador ssPitch2U
eig4=eig(ssPitch2U.a);
VAPCont4=-2;
K4=place(ssPitch2U.a,ssPitch2U.b,[VAPCont4])

% Controlador ssYaw
eig5=eig(ssYaw.a);
VAPCont5=-2;
K5=place(ssYaw.a,ssYaw.b,[VAPCont5])

% Controlador ssH
eig6=eig(ssHM.a);
VAPCont6=[-2 -2.01];
K6=place(ssHM.a,ssHM.b,[VAPCont6])
```

A.3 Càlcul de l'observador d'estat

```
% Observador ssRoll
eig1=eig(ssRoll.a);
VAPObs1=eig1*10;
L1=(place(ssRoll.a',ssRoll.c',VAPObs1))';

% Observador ssRoll2V
eig2=eig(ssRoll2V.a);
VAPObs2=eig2*10;
L2=(place(ssRoll2V.a',ssRoll2V.c',VAPObs2))'

% Observador ssPitch
eig3=eig(ssPitch.a);
VAPObs3=eig3*10;
L3=(place(ssPitch.a',ssPitch.c',VAPObs3))';

% Observador ssPitch2U
eig4=eig(ssPitch2U.a);
VAPObs4=eig4*10;
L4=(place(ssPitch2U.a',ssPitch2U.c',VAPObs4))'

% Observador ssYaw
eig5=eig(ssYaw.a);
VAPObs5=eig5*10;
L5=(place(ssYaw.a',ssYaw.c',VAPObs5))'

% Observador ssH
eig6=eig(ssHM.a);
VAPObs6=10*[-5.82 -5.83];
L6=(place(ssHM.a',ssHM.c',VAPObs6))'
```

A.4 Càlcul dels adequadors de consigna

```
% Adequador de consigna per a la velocitat lateral
% Transformarà el valor desitjat de V en Roll Ref.
adq1=inv(-ssRoll.c/(ssRoll.a-ssRoll.b*K1)*ssRoll.b)

% Adequador de consigna per a la velocitat lateral
% Transformarà el valor desitjat de V en Roll Ref.
adq2=inv(-ssRoll2V.c/(ssRoll2V.a-ssRoll2V.b*K2)*ssRoll2V.b)

% Adequador de consigna per a la velocitat frontal
% Transformarà el valor desitjat de U en Pitch Ref.
adq3=inv(-ssPitch.c/(ssPitch.a-ssPitch.b*K3)*ssPitch.b)

% Adequador de consigna per a la velocitat frontal
% Transformarà el valor desitjat de U en Pitch Ref.
adq4=inv(-ssPitch2U.c/(ssPitch2U.a-ssPitch2U.b*K4)*ssPitch2U.b)

% Adequador de consigna per al Yaw
% Transforma el valor desitjat de Yaw en Yaw Rate Ref.
adq5=inv(-ssYaw.c/(ssYaw.a-ssYaw.b*K5)*ssYaw.b)

% Adequador de consigna per a l'altura
% Transforma el valor desitjat de H en Vrt. Spd. Ref.
adq6=inv(-ssHM.c/(ssHM.a-ssHM.b*K6)*ssHM.b)
```